Mind The Gap: Newsfeed Visualisation with Metro Maps

Damask Talary-Brown

Bachelor of Science in Computer Science with Honours
The University of Bath
2017

This dissertation may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation.

Signed: 

Mind The Gap: Newsfeed Visualisation with Metro Maps

Submitted by: Damask Talary-Brown

COPYRIGHT

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed: 

Abstract

News overload has emerged as a growing problem in our increasingly connected digital information era. With complex long-running stories unfolding over weeks and months, young adults in particular are left overwhelmed and demotivated, which leads to their disengagement from politics and current events news.

This dissertation presents a method for the automatic generation of metro maps based on news content obtained from user-specified RSS feeds. Metro maps are familiar to most adults, and they are intuitive visual metaphors for representing concepts which branch and diverge, such as news stories. The method described performs entity disambiguation and various other NLP techniques to extract a set of topics (*metro lines*) from a news corpus which provide a cohesive summary of its content.

The difficulty of drawing unoccluded octilinear metro maps is a barrier to their current utility in InfoVis. Therefore, this dissertation also introduces a heuristic force-directed approach for drawing metro maps, which is refined using multicriteria optimisations taken from neighbouring literature in information cartography.

The resultant system is demonstrated using the RSS feeds published by several popular British newspapers, and empirically evaluated in a user study. The results of the study support the hypothesis that metro map users demonstrate greater topic recall than users of an equivalent RSS reader. Lastly, areas for future research are discussed, followed by recommendations for the commercial development of this and similar systems.

Contents

List of Figures

List of Tables

Acknowledgements

Firstly, I'd like to thank Professor Stephen Payne for his supervision, advice, and cheerful weekly repetition of the phrase "Just carry on!" Without his guidance and encouragement, this project would not have been possible.

I also have to thank my parents for their unending love and support, as well as the sacrifices they've made over the years for me and my education.

Last but not least, thank you to all my of friends both in Bath and in London, for keeping me sane this year, for letting me ramble about tube maps for hours, and for the constant supply of sweets.

Introduction

"The Press, Watson, is a most valuable institution, if you only know how to use it."
— Sherlock Holmes, *The Adventure of the Six Napoleons*
Sir Arthur Conan Doyle

Why don't we Understand the News?

The day the result of the 2016 United Kingdom EU Membership Referendum was announced, the @GoogleTrends Twitter account reported a 250% increase in searches for "What happens if we leave the EU?" Much like the case of David Leonhardt's 2008 article in the New York Times which began, "Raise your hand if you don't quite understand this whole financial crisis," national news commentary had focused on little else in the preceding months.

Some months after Leonhardt's article was published, Journalism Professor Jay Rosen voiced his agreement with its premise in a blog post on the failure of journalism during the financial crisis; "there are certain very important stories – and the mortgage crisis is a good example – where until I grasp the whole I am unable to make sense of any part."[Rosen, 2008]

Recent studies have found that the general public use news media for many purposes; to make important life decisions, for entertainment, as a requirement of their jobs, and out of perceived civic obligation [Shahaf et al., 2015, Purcell et al., 2010]. As a result of the ongoing shift towards online multimedia journalism, there has been an explosion of globally accessible knowledge which is expanding at an unprecedented rate as the internet grows.

Although ubiquitous news media has resulted in more immediate and diverse coverage of current events, the volume of content available online has made the process of understanding it both daunting and off-putting to young adults [Associated Press and Context-Based Research Group, 2008]. Additionally, while news aggregators and RSS readers make it faster for users to access the news they care about, they do not aid comprehension. In spite of these facts, little attention has been given to addressing the problem that understanding news articles individually is inherently reliant on understanding news articles as a collection.

Existing information infrastructure has been criticised both for not supporting the cross-correlation between collections of related news articles [Rennison, 1994], and for attempting fit complex narratives into reductive and misleading visualisations [Shahaf et al., 2015]. Research into how humans' spatio-cognitive abilities can be applied to more abstract visual metaphors [Skupin, 2000] suggests the strength of visual metaphors lies in their familiarity. We therefore build on the work of Shahaf et al. [2012*b*] to integrate the Metro Map metaphor into the news aggregation process.

Key Aims and Contributions

The primary aim of this project is the development of a tool which generates interactive metro maps of news from RSS feeds, with individual articles transformed into *stations* and common themes transformed into *metro lines*. Our goal is to reduce the information overload experienced by news consumers by providing contextual links between articles and topics.

The resultant system is a news feed aggregator with graphically structured output, and to the best of our knowledge is first of its kind. Contributions of this dissertation include:

- Implementation of efficient keyword extraction using tf-idf [Sparck Jones, 1972] and tf-pdf [Bun and Ishizuka, 2002] on a keyword space of named entities (Section 3.3).
- A novel and lightweight method for performing entity disambiguation within current events articles using Google's Knowledge Graph API (Section 3.3.1.2).
- Formalisation of the metrics *Line Coverage* and *Affinity* as criteria for evaluating and ranking candidate metro lines (Sections 3.4.1 and 3.4.2).
- Recommendations for how D3.js parameters can be manually tuned to generate initial force-directed station positions for planar embeddings of metro maps (Section 3.5.2).
- The first application of Stott's [2008, 2011] aesthetic criteria for metro maps to maps drawn from news corpora (Section 3.5.1).
- An empirical evaluation which provides statistical evidence to support the hypothesis that users recall more topics after using our metro maps than after reading news structured as a chronological list (Chapter 6).

Outline

The structure of this dissertation is as follows:

Chapter 1 provides an overview of the background literature upon which this project relies, including an introduction to information overload and sensemaking, and a justification of appropriate visualisations for news corpora.

Chapter 2 describes the scoping of the system, the informal requirements gathering process undertaken, and the rationale behind the significant design decisions made.

Chapter 3 discusses the algorithms and techniques chosen to transform the data from feed to visualisation, describes how they were implemented in the system and explains the trade-offs which were encountered.

Chapter 4 provides a set of example results generated by the system from different RSS feeds and analyses them from the perspectives of content, layout, and context.

Chapter 5 describes the process by which we evaluated the system in a two-part user study and discusses the results and implications of our experiments.

Chapter 6 discusses both the contributions and limitations of this work, and provides a discussion on future research directions.

Chapter 1
Literature Review

1.1 News Overload as a Design Problem

It has become apparent that prolific coverage alone is not enough to engage and support the public in understanding the complexities of current events. Historically, news media has been limited in the volume of content it can produce by physical constraints such as printing costs, but the rise of the internet as a platform to deliver the news has lead to an explosion of content, both through existing media channels and through competing social media websites and blogs. News is no longer information we have to actively seek out; it is inescapable.

The term *ambient news* was coined by Hargreaves et al. [2002] to describe the ubiquity of news in the current information landscape. Others journalists have commented on the issue in a more critical light, describing the proliferation of competing news media as "as pervasive–and in some ways as invasive–as advertising." [Nordenson, 2008, p.2]

In 2008, The Associated Press conducted an extensive field study into the news consumption habits of young adults. Among their key findings were three points which acutely summarise the news overload problem;

- **"Consumers are experiencing news fatigue."**

  The study found participants were debilitated and overwhelmed, and that their levels of dissatisfaction lead to a decrease in the effort they put into news acquisition. This is consistent with multiple other studies [Holton and Chyi, 2012, Purcell et al., 2010, Fischer and Stevens, 1991] which found participants across every demographic were overwhelmed by the amount of news content available to them and agreed that it prevented them exploring news on less familiar topics.

- **"Story resolution is key."**

  Participants' consistent enjoyment of sports and entertainment news was due in part to the formulaic storytelling which characterises these types of journalism, with clear chronology to provide contextual back story. The feeling of enjoyment gained from reading procedural stories directly contrasts with what the same participants experienced reading World news, where they struggled to find resolution and context for stories which were unfolding at the time.

- **"Consumers want depth but aren't getting it"**

  It was observed that participants, in their efforts to discover *below-the-fold* content (defined in the context of the AP's model, 2008, p.37) from particular headlines, often found themselves reading the same summary-level content from different news sources. It was recommended that news providers support this by "designing innovative formats and creating easier pathways to deep content." [Associated Press and Context-Based Research Group, 2008, p.49]

Initially, the third point seems to be a direct contradiction to the first; we are overwhelmed by the volume of news we are exposed to, but we also crave more detail from the news we do consume. However, it brings to light the issue of information *quality* as a requirement of news consumers.

Journalism, and therefore journalistic quality, can be viewed along a spectrum between two models; a model for the communication of facts, and a model for entertainment and storytelling. From the three points above, it is apparent that quality at both ends of the spectrum is being sought, since the desire for quality below-the-fold content is covered by the first model, and the desire for quality story resolution by the second.

1.1.1 Information Overload

News fatigue is a domain-specific type of information overload, a phenomenon formally defined as "when the information processing demands on time to perform interactions and internal calculations exceed the supply or capacity of time available for such processing" [Schick et al., 1990, p.206]. Information overload is a multifaceted problem which can be modelled as a combination of three contributing factors (Figure 1.1).

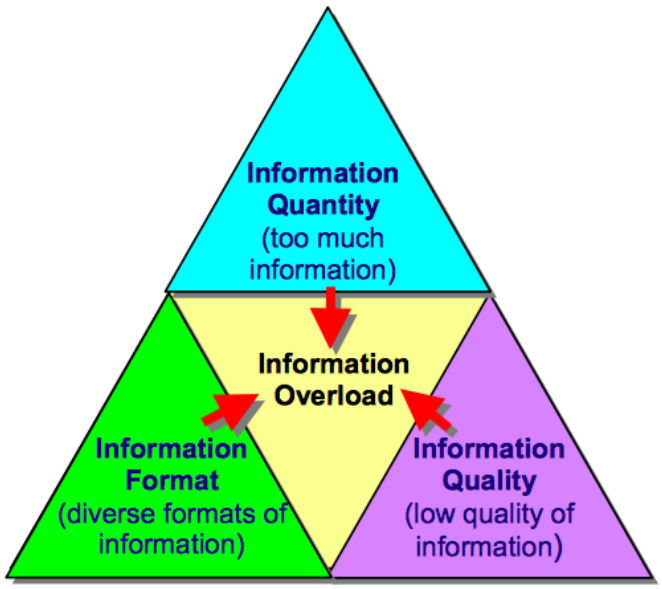


Figure 1.1: Dimensions of information overload, as defined by Ho and Tang [2001].

These factors correspond directly to the three points previously identified from the Associated Press and Context-Based Research Group study. High information quantity leads to news fatigue, information format determines the level of possible story resolution, and information quality determines how much depth a reader can gain from the news they consume. The authors did not find a single solution which could address all three factors, but they did identify information quantity as the most significant contributor to overload.

Bergamaschi et al. [2010] further decompose information quantity into spatial and temporal dimensions in the context of news articles specifically. Spatial quantity refers to articles which are near-identical in terms of facts presented being published by different media outlets, and temporal quantity refers to articles on a single topic being published in quick succession over a short period of time.

Intuitively, the terms spatial and temporal quality seem illogically named, as a set of articles with high spatial quantity would cover a smaller area of information space and vice versa. High spatial quantity will therefore be referred to as *redundancy*, and high temporal quantity as *fragmentation*, since a sudden burst of articles published on the same topic suggests a currently unfolding story being told in parts.

To adequately determine the contributory factors relating to news fatigue, all four dimensions of information overload should be considered, and will therefore be explored in more detail in the following sections.

1.1.1.1 Information Quality

In the context of factual data rather than news specifically, Strong et al. [1997] defines information quality in terms of four components; intrinsic quality, accessibility quality, contextual quality and representational quality. If the news can be reduced to its core function as an interpretation of facts and other raw data such as images, then this same framework can be experimentally applied to news journalism in order to determine which factors could influence its quality.

Intrinsic quality is a measure of the accuracy, objectivity, believability and reputation of data. In the context of news, the first three factors would typically be true for all major news sources, and the reputation would be dependent on whether the article originated from a trusted source or not. Accessibility quality is less relevant to online news media, as it is concerned with data access and security. Contextual quality is the most relevant category in respect to news, concerning timeliness, amount of data, and value-added. In the news domain, this would mean an article's quality is dependent on its performance against a background of other articles; whether or not it contributes anything recent or previously unknown. Finally, representational quality is concerned with ease of understanding and interpretability, which are easily translatable concepts.

The implications of applying the Information Quality Framework [Strong et al., 1997] to news articles are that quality may be influenced by the reputation of the source, the timeliness of publication, value-added by the article (i.e. content which couldn't be derived from other sources) and the ease of understanding of the content.

1.1.1.2 Fragmentation (Temporal Quantity)

The rise of social media websites such as Twitter delivering news to consumers has lead to a high degree of news fragmentation, due to the constraints of the microblogging service's 140-character limit. 24-hour television news paved the way for new formats of real-time content delivery, and the ever-expanding network of online social media channels followed.

It is logical due to the fragmented nature of real-time news journalism that temporal quality suffers; stories are published and updated intermittently over short periods of time, meaning there is more content for the consumer to piece together in order to understand a story. The fragmentation is somewhat mitigated by Twitter's use of hashtags to annotate a Tweet with its topics. Hashtags help readers form a coherent picture of unfolding events from the incremental contributions of thousands of participating users [Bruns et al., 2012].

Phuvipadawat and Murata [2010] developed a methodology to collect and group Tweets on breaking news topics, using hashtags for topic identification or *story-finding*, and grouping similar messages together to form a single news story. Their algorithm for similarity is a function of the tf-idf [Salton and Buckley, 1988] of the two messages and the number of named entities they have in common.

1.1.1.3 Redundancy (Spatial Quantity)

It is in the nature of news that newsworthy stories get repeated across multiple sources. When consumers read news on a particular event from more than one source, it is likely that they will read variations on the same facts in multiple articles.

Attempts such as [Barzilay et al., 1999] have been made to synthesise summaries of collections of similar online documents, a practice here termed *information fusion*, with news articles from different sources being given as a specific use-case. However, the process of extracting common sentences between documents was in order to reformulate them into a single summary, rather than to determine the level of similarity between the documents.

A more relevant approach was presented by Pera and Ng [2008], who used the *title* and *description* attributes of elements in RSS feeds as content descriptors to mitigate the overhead of processing entire documents for phrases. The content descriptors are then used to compute phrase n-grams as a measure of similarity between any two documents. The similarities in this case were used to remove subsumed articles and cluster non-redundant similar ones, in order to streamline feed content for readers.

It should be noted that there is an overlap between the notion of spatial quantity and one of the four influencing factors in information quality; contextual quality. If a feed contains two articles which state the same number of identical facts, they therefore contribute to information overload on both the qualitative and quantitative fronts.

1.1.1.4 Information Format

The domain of news articles is a more specific information space than that of traditional documents in general, and by nature most news articles share some common formatting and

structural elements such as headlines, timestamps, and relevant images. As a result of this, it is unlikely that any two articles from popular news providers would be diverse enough in content format to overshadow the information quantity problem.

Viewing the other dimensions of overload from a news domain perspective, it is clear that (consistent with the findings of Ho and Tang [2001]) information quantity is the most relevant contributing factor in respect to fatigue, alongside factors influencing contextual quality such as value-added and timeliness. Any proposed solutions to the news overload problem should therefore address these factors first.

1.1.2 Supporting Sensemaking

Sensemaking is the basis for forming contextual knowledge; the process by which we incorporate new information into our existing cognitive frameworks, and how we go from reading something to understanding it [Pentina and Tarafdar, 2014]. In broader terms, Weick et al. describe sensemaking as "[being] about the question: What does an event mean? In the context of everyday life, when people confront something unintelligible and ask, 'What's the story here?' " [Weick et al., 2005, p.85]

This definition relates directly to the news overload problem because one component of sensemaking is the contextual story resolution that The Associated Press and Context-Based Research Group [2008] study identified news consumers are craving. It has also been observed that often readers are not interested in specific articles on a subject, and only the thematic content of the topic they belong to [Husin et al., 2014]. How then, do readers make sense of a collection of articles surrounding a particular topic?

When presented with a large document collection, Russell et al. [2006] found all of their subjects began by clustering the contents into groups which formed a heuristic representation or mental model, used to provide an overview. However, current information infrastructure has been criticised for not supporting the cross-correlation between connected news articles [Rennison, 1994].

Writing for the Columbia Journalism review in 2008, Nordenson outlined a suggestion for the new role of journalism in the information era; "By linking stories to one another and to background information and analysis, news organizations help news consumers find their way through a flood of information that without such mediation could be overwhelming and nearly meaningless." [Nordenson, 2008, p.10]

Similarly, Pentina and Tarafdar [2014] recommend in the context of contemporary media consumption that news providers should adapt to an environment of news overload by adding facilities enabling readers to categorise, sort and search news collections. Additional findings of this study suggested that the contextual background provided by having more detailed coverage aids the sensemaking process, as it helps users form links between new information and their existing frameworks, but this presents an interesting conflict with the goal of reducing information overload when considering large collections of documents.

It is apparent that many recommendations have been made from within the field of journalism that at the point of delivery, news content should incorporate contextual links between related articles. This is important both from a sensemaking perspective to emphasise con-

nections, and from an information overload perspective to help users find meaning in an inundated news landscape.

The news overload problem can now be reformulated with scope and detail: How can we display a collection of related news articles in such a way that users are not overwhelmed by unstructured content and are free to explore the underlying contextual pathways?

A simple starting point comes from a familiar idiom; a picture is worth a thousand words.

1.2 An Introduction to InfoVis

Of course, a picture is not always worth a thousand words, particularly when the picture is unstructured and complex in its own right. However, a recognised and effective technique for bridging the gap between a set of data and a user's mental model and subsequent comprension of the data is information visualisation, or InfoVis. [Yi et al., 2008, Havre et al., 2002]. This section provides a brief overview of a formative InfoVis taxonomy and uses this taxonomy to categorise appropriate visual models for newsfeed visualisation.

In his seminal paper on information visualisation, Shneiderman proposed a taxonomy for visualisations comprising seven data type abstractions, and seven tasks which are components of the visual information seeking mantra; "Overview first, zoom and filter, then details-on-demand." [Shneiderman, 1996, p.1]

Shneiderman's type abstractions are as follows:

1-dimensional	Linear data, where each datum is a string of characters.
2-dimensional	Planar data, e.g. layout diagrams, or document clusters.
3-dimensional	Physical objects or models of real-world entities, e.g. computer aided designs or medical imaging data.
Multi-dimensional	Any data where items with n attributes can be represented in n-dimensional space, e.g. relational databases, or feature vectors for classification.
Temporal	Data following a timeline, which is a subset of 1-dimensional data but was deemed important enough to warrant its own category. E.g. Project management data, or multimedia content timelines.
Tree	Hierarchical data where each datum has exactly one parent and zero or many children, e.g. document or directory structures.
Network	Related data, where each datum can have an arbitrary number of links to other data.

Because of the non-spatial nature of textual data, any visualisation of such data must involve some form of content abstraction and translation into a physical space [Wise et al., 1995]. These translations can result in data of arbitrary dimensionality, so a text corpus could fall into the 1-dimensional or multi-dimensional categories. News articles as a specific subset

of textual data have certain metadata associated with them including dates, meaning they also fit the temporal type abstraction. In addition to this, if contextual links are considered part of the structure of the data, articles can be modelled as a network of connected nodes.

This ambiguity is not a failure of the taxonomy; Shneiderman stresses that composite categories are equally valid. However, the implications of this are that the most appropriate visualisation for a news corpus may itself be a composite of visualisations for any of its type abstractions, leaving an unfeasible number of possibilities to consider.

To reduce the scope of suitable visualisations, we return to the original problem of information overload. This time however, the aim is to minimise the overload from interpreting the model, in addition to reducing the overload from interpreting the data. Complex visualisations which require a considerable amount of effort to understand in their own right should be avoided when reducing overload is the goal.

1.2.1 InfoVis for Sensemaking

In addition to the insight that visualisation may be able to provide, there is evidence that visual metaphors better support the learning process and are more easily remembered than isomorphic text representations alone [O'donnell et al., 2002, Yen et al., 2012].

Yi et al. [2008] identified four overlapping InfoVis processes which describe how insight can be gained after sensemaking; *Provide Overview*, *Adjust*, *Detect Pattern*, and *Match Mental Model*. These four processes can be roughly mapped to Shneiderman's high-level tasks, which are as follows:

Overview	Gain a birds-eye view of the entire collection, with the option to change the scale of the view by zooming or using fisheye magnification techniques.
Zoom	Gain a more detailed view of a portion of data or single datum while preserving the original sense of context.
Filter	Nondestructively remove uninteresting data points or groups from the view.
Details-on-Demand	Gain additional insight into one or more data points by selecting particular elements.
Relate	View and explore relationships between elements.
History	If necessary, undo actions to return to a previous view of the data.
Extract	Export selected data, preserving the format, for uses such as "sending by email, printing, graphing, or insertion into a statistical or presentation package" [Shneiderman, 1996, p.5].

The *Provide Overview* process allows a reader to recognise what they know and what they don't know from the information they are processing. The corresponding task in [Shneiderman, 1996] is *Overview*.

Adjust allows them to change the level of abstraction or field of selection of that information. This corresponds to *Zoom* and *Filter* in Shneiderman's task model.

The *Detect Pattern* procedure is where structure and trends are found (whether expected or otherwise). Coupled with *Match Mental Model*, where the links are formed between the new data and the users' existing cognitive frameworks, this corresponds to *Relate*.

At this point, Shneiderman's taxonomy diverges from the processes of Yi et al., who are concerned with the cognition enabled by visualisation. In contrast, Shneiderman considers additional use cases for visualisations, such as querying and sharing.

From these two models, it is apparent that certain views and functions are crucial for tools which use visualisation to support the sensemaking process; a high-level overview visualisation which emphasises links between data, the ability to adjust scope to show more or less detail, and the ability to filter information of specific interest within the dataset.

1.2.2 Visual Metaphors

Eppler defines visual metaphors as "a graphic structure that uses the shape and elements of a familiar natural or man-made artefact [...] to organize content meaningfully and use the associations with the metaphor to convey additional meaning about the content." [Eppler, 2006, p.203] This definition highlights the main advantage to using visual metaphors; users are intuitively familiar with how they present and structure information.

The use of preexisting visual metaphors–specifically those with which a large number of people will already be familiar–has been shown to support readers' comprehension, as it requires both significant time and effort for a reader to interpret visual metaphors which are new to them [Ziemkiewicz and Kosara, 2009].

In a previous paper, Eppler also describes six advantages of visual metaphors specifically for the transfer of knowledge: "(1) to motivate people; (2) to present new perspectives; (3) to increase remembrance; (4) to support the process of learning; (5) to focus the attention of the viewer and (6) to structure and coordinate communication." [Burkhard, 2004, p.2, citing [Eppler, 2004]]. These are all desirable attributes, but motivation and support in the learning process are particularly relevant in addressing news fatigue.

Examples of visual metaphors commonly used to represent collections of data include calendars, bookshelves, timelines, maps and other schematics. Metaphors based on physical objects do not have to be visually skeuomorphic to be effective, but to avoid misinterpretation, there should be a match between the underlying structure of the metaphor and the underlying structure of the data. Two common and highly structured visual metaphors will be explored in the following sections in the context of potential for news representation; timelines and schematic maps.

1.2.2.1 Timeline Visualisations

Chronological ordering is an important characteristic of news articles and should be preserved in any visualisation of news data as it provides a natural ordering [Binh Tran, 2013]. Perhaps the simplest visual metaphor for a collection of dated documents is the timeline.

Nguyen et al. [2014*b*] explore the role of timelines in the sensemaking process, emphasising that the interactions supported by such visualisations should be as intuitive as possible in order to not disrupt users' trains of thought, and should be tightly coupled with other elements of the sensemaking process so temporal connections are not viewed solely in isolation.

Criticisms of previous timeline visualisations made by the authors are that linear layouts are often too simple for the data they represent, and that a lack of automatic layout generation results in additional manual work for the user.

The colouring technique used to distinguish sets of related events within a single timeline is a flexible extension to [Nguyen et al., 2014*a*], where the authors coloured events belonging to multiple sets with a gradient composed of the colours of both sets. However, the gradient approach presented in [Nguyen et al., 2014*b*] does not scale to events belonging to more than two sets, since the colour grouping restricts the number of possible intersections of each set. From a news storyline perspective, this would place an upper limit of two on the number of possible topics a story could belong to, which is a low constraint for all but the highest level topics.

Singh et al. [2015] designed a prototype for generating annotated timelines based on the Wikipedia entries long-running news stories. The use of Wikipedia rather than newsfeeds meant their document retrieval model was heavily dependent on Wikipedia's structure, but it also afforded a huge wealth of contextual information that made such detailed annotations possible. Not all stories are long running however, so while this would be useful as a retrospective tool it would be impossible to generate timelines in the same way for news articles which did not already belong to a long-running chain of events.

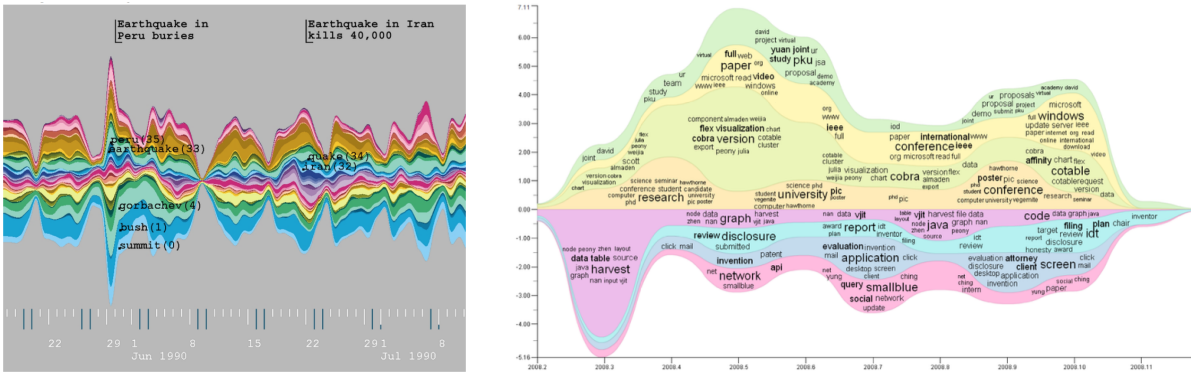


Figure 1.2: Similar visualisations from ThemeRiver [Havre et al., 2002] (left) and TIARA [Liu, Zhou, Pan, Qian, Cai and Lian, 2009] (right).

Both ESTHETE [Goyal et al., 2013] and nReader [Wang et al., 2006] present timeline-centric views for collections of news articles based on underlying graphs of relationships between the articles. However, in both cases, the graph structure was not part of the final visualisation, so connections between entities were displayed in purely textual forms. In contrast, ThemeRiver [Havre et al., 2002] introduces a novel view on topic frequency along the time axis to show thematic change over time within a collection of documents, similar to a smoothed histogram. This view, while useful for large document collections which span weeks or months, would be less suited to displaying emerging news trends over shorter periods.

TIARA [Liu, Zhou, Pan, Qian, Cai and Lian, 2009] whose authors cite ThemeRiver as an influencing design (see Figure 1.2), displays a similar shaped graphical output but performs more detailed textual analysis, and displays related keywords in the output. Both visualisations support simple zooming and panning, but suffer from the same limitations on visualising topic connectivity as [Nguyen et al., 2014*a*].

Taking into consideration both the critiques of oversimplification identified in Nguyen et al. [2014*b*] and the physical limitations highlighted in Nguyen et al. [2014*a*] and Havre et al. [2002], it is clear that timelines are not the most appropriate visual metaphor for highly connected events and news topics. However, for more linear storylines which span fewer categories or topics, a visualisation such as [Nguyen et al., 2014*b*] could be used, for example as part of Shneiderman's *Zoom and Filter* task where the dataset is pruned.

1.2.2.2 Topological and Schematic Map Visualisations

The use of cartographic representations for abstract objects and the relationships between them is such a common and natural metaphor that it is often unnoticed in digital contexts. Maps take advantage of humans' natural ability to perceive and organise in a spatial context, and the fact that we live in a spatial world "leads naturally to metaphors that provide cues for orientation and navigation." [Old, 2002, p.2]

Topological maps[1] are a specific class of map which abstract away detail so that only significant features within desired subsets of the mapped dataset remain; from a news overload perspective, this is a highly desirable property. These maps are often used to visualise networks and are represented as schematics, where elements on the map are transformed into abstract visual representations for ease of understanding. Today, the most recognisable examples of topological maps are transit maps. Divorced from the strictness of geographical accuracy and scale, emphasis is instead placed on the usability of the map for planning and the understanding of relative positioning which the maps enable [Hochmair, 2009].

Figures 1.3 and 1.4 show the geographically accurate 1930 London Underground map designed by Frederick Stingemore, and the iconic 1933 redesign by Henry Beck, which was based on the concept of an electrical circuit diagram [Transport For London, 2014]. While Beck's simplification of the structure of the map was seen as radical and even controversial at the time, the hallmarks of his design are now recognisable not just in other schematic maps, but in posters, infographics and diagrams across various domains.

In terms of usability, simple timeline visualisations can provide the chronological ordering which is intrinsic within collections of news articles. However, schematic maps may be able to represent complex relationships between topics in a structure which is linear but has an additional dimension, to prevent linearity becoming a design or usability constraint.

[1] Not to be confused with *topographic* maps, which represent relief and other geographic features of physical regions at a large scale and in fine detail.

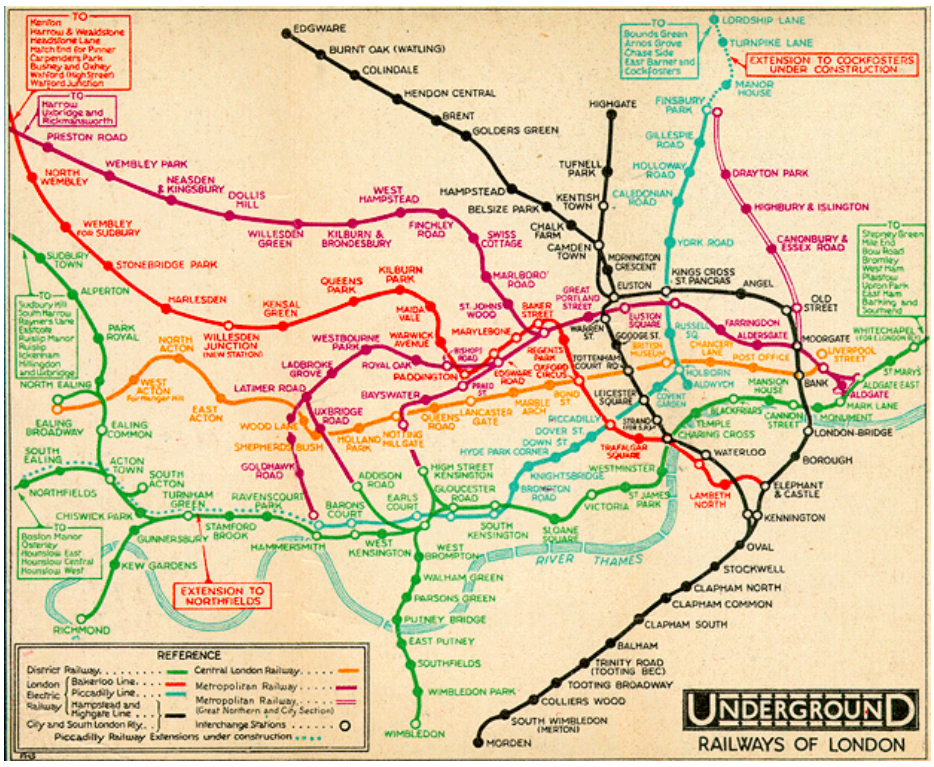


Figure 1.3: 1930 London Underground map, designed by F. Stingemore [British Broadcasting Corporation, 2013].

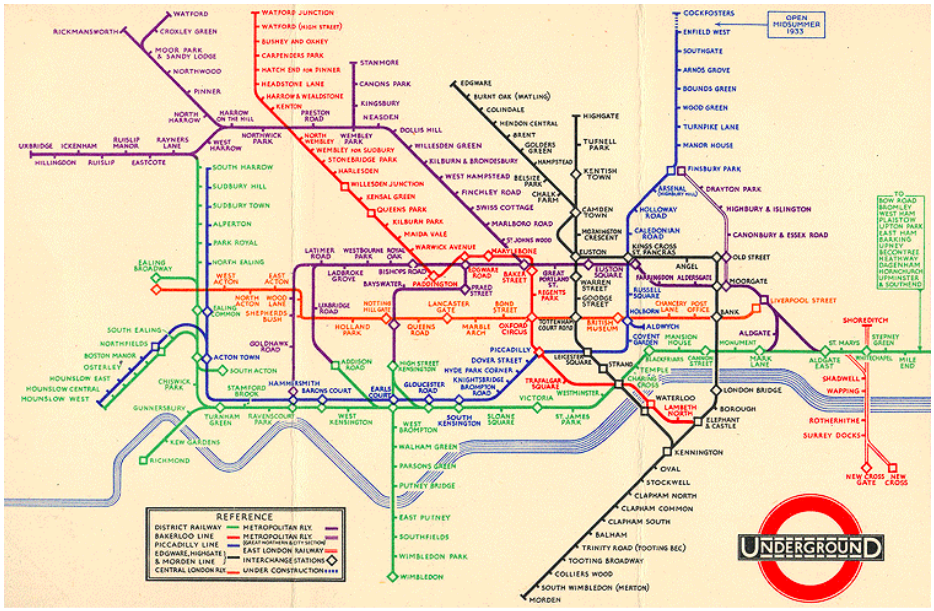


Figure 1.4: 1933 London Underground Map, designed by H. Beck [British Broadcasting Corporation, 2013].

1.3 Metro Maps for Information Cartography

Significant work in the area of information cartography has been undertaken by Shahaf et al. [Shahaf and Guestrin, 2010, Shahaf et al., 2012*b*,*a*, 2013], in the domains of both news and science, through the visualisation of article and journal data on metro maps. The authors chose the metaphor as a base for their visualisations to address the fact that previous timeline-based summarisation systems could only represent simple linear stories; "In contrast, complex stories display a very non-linear structure: stories split into branches, side stories, dead ends, and intertwining narratives." [Shahaf et al., 2013, p.1]

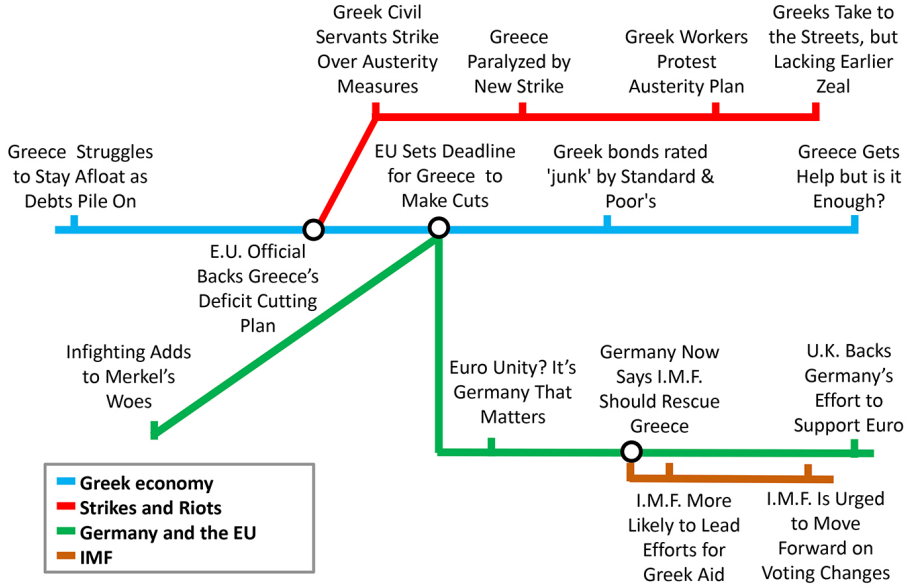


Figure 1.5: A metro map [Shahaf et al., 2012*b*] covering the Greek Debt Crisis.

Even in an academic context, this was not the first time the abstract visualisation potential of the metro map had been noted; "The usefulness of the metro map as a metaphor is somewhat limited to simple examples by the time required to manually produce these maps. As such they are generally only useful for applications that do not change frequently. This limitation could be removed by quality methods for the automatic drawing of metro maps from abstract data." [Stott, 2008, p.54]

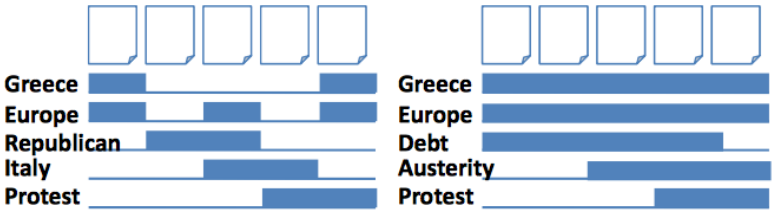
In this section, the formalisation of the metro map metaphor, its associated characteristics, and its limitations will be discussed.

Definition 1. *Metro Map [Shahaf et al., 2012*b*]: A metro map* $\mathcal{M}$ *is a pair* (G, Π)*, where* $G = (V, E)$ *is a directed graph and* Π *is a set of paths, or metro lines in* G*. Each* $e \in E$ *must belong to at least one metro line.*

A previously published method [Shahaf and Guestrin, 2010] for linking together chains of articles was discussed, and an objective function was created to formalise the characteristics of a 'good' metro map. The function defined was a composite based on three important characteristics, all of which are broadly applicable to the visualisation of any similar corpora; coherence, coverage, and connectivity.

1.3.1 Coherence

Let $\mathcal{D}$ be a set of articles, and $\mathcal{W}$ be a set of words or phrases, such that each article is a subset of $\mathcal{W}$. A *coherent* chain of articles through $\mathcal{D}$ is one where transitions between documents are smoothed by common overlapping keywords from $\mathcal{W}$, creating a better narrative flow [Shahaf and Guestrin, 2010] as depicted in Figure 1.6.



A bar corresponds to the presence of a word in the article above it.
The titles of the articles which made up the two chains were as follows:

Chain A (left)	Chain B (right)
Europe weighs possibility of debt default in Greece	Europe weighs possibility of debt default in Greece
Why Republicans don't fear a debt default	Europe commits to action on Greek debt
Italy; The Pope's leaning toward Republican ideas	Europe union moves towards a bailout of Greece
Italian-American groups protest 'Sopranos'	Greece set to release austerity plan
Greek workers protest austerity plan	Greek workers protest austerity plan

Figure 1.6: An incoherent chain with jittery transitions between topics (Chain A, left) alongside a more coherent chain of articles (Chain B, right). [Shahaf et al., 2012*b*]

Coherence, intuitively, seems to be closely linked to idea of story resolution detailed in Section 1.1.1. This presents a question which could later be explored further; does forming coherent chains of articles provide the story resolution that participants in the Associated Press study were so desperately seeking from current events journalism?

1.3.2 Coverage

As in the previous section, let $\mathcal{D}$ be a set of articles, and $\mathcal{W}$ be a set of words or phrases of which the articles are composed. The coverage function for a word in a given document $d_i \in \mathcal{D}$ specified in Equation 1.1 can be quantified using any measure of how well d_i covers w, for example tf-idf$(w, d_i, \mathcal{D})$ (See Equation 1.6) [Shahaf et al., 2012*b*].

$$cover_{d_i}(w) : \mathcal{W} \rightarrow [0, 1] \tag{1.1}$$

Extending the notion of coverage to maps–which can be abstracted to sets of documents–introduces the idea of *diversity*. If a map already contains documents which for a sufficient coverage for some word w, then there is nothing to be gained by adding another document to $\mathcal{D}$ which has high coverage of w alone. This relates back to the principles of spatial and information quality discussed in Section 1.1.1, especially the importance of value-added by every individual document in a collection. In this case, maps which cover a maximal number of $w \in \mathcal{W}$ should be preferential. A simple additive definition for map coverage

such as Equation 1.2 [Shahaf et al., 2012*b*] would not reward this kind of diversity;

$$cover_{\mathcal{M}}(w) = \sum_{d_i \in docs(\mathcal{M})} cover_{d_i}(w) \tag{1.2}$$

Therefore, an alternative definition for map coverage was chosen, which will not increase significantly if another document which covers an already covered feature is added to $\mathcal{D}$ (Equation 1.3 [Shahaf et al., 2012*b*]).

$$cover_{\mathcal{M}}(w) = 1 - \prod_{d_i \in docs(\mathcal{M})} (1 - cover_{d_i}(w)) \tag{1.3}$$

Finally, the definition of map coverage is extended to the coverage of the corpus $\mathcal{D}$, rather than just single features. If each feature is weighted according to frequency, then for each $w \in \mathcal{W}$ we have some λ_w. The coverage of a corpus $\mathcal{D}$ by a metro map $\mathcal{M}$ can then be defined as in Equation 1.4 [Shahaf et al., 2012*b*].

$$Cover(\mathcal{M}, \mathcal{D}) = \sum_{w \in \mathcal{W}} \lambda_w cover_{\mathcal{M}}(w) \tag{1.4}$$

1.3.3 Connectivity

The final property is the most simply defined; the connectivity of a metro map is the number of paths in Π which intersect [Shahaf et al., 2012*b*].

$$Connectivity(\mathcal{M}) = \sum_{i<j} \mathbb{1}(p_i \cap p_j \neq \emptyset) \tag{1.5}$$

1.3.4 Limitations of Shahaf et al. [2012*b*, 2013]

1.3.4.1 Corpus

Perhaps the biggest limitation of the system developed by Shahaf et al. is the nature of the corpus $\mathcal{D}$; it is a fixed dataset, meaning users can only query it for certain past events with no way of specifying a different corpus themselves.

From a historical reference perspective the output generated based on certain queries is interesting when compared to the output an expert would select as important to the narrative, but it is not possible to use the system as a replacement to a newsfeed aggregator.

1.3.4.2 Graph Layout Aesthetic Principles

From a usability perspective, a second limitation of the work of Shahaf et al. is the lack of focus given to the desirable aesthetic properties of transit maps.

A formative empirical study [Purchase et al., 1997] on how graph layout affects usability identified a collection of five measurable aesthetic principles from previous research, which aid human understanding when reading graphs:

- Minimise the number of line bends [Tamassia, 1987];
- Minimise the number of edge crossings [Ferrari and Mezzalira, 1970];
- Preserve any underlying symmetry in the structure of the graph [Lipton et al., 1985];
- Draw orthogonally where possible [Tamassia, 1987];
- Maximise the minimum angle between incident edges for each node [Garg and Tamassia, 1994].

These criteria, when numerically calculated and weighted relative to one another, allow the aesthetic quality of any graph to be evaluated. This is particularly useful for graphs generated by automatic layout algorithms, which aren't as much a product of human design intuition as their manually designed counterparts.

The origins of the principles above predate much of the early InfoVis research because several of them [Tamassia, 1987, Ferrari and Mezzalira, 1970] were formalised as guidelines for the design of electronic circuits.

In a later study, Purchase evaluated information-finding task performance with a set of graphs which varied the above principles, to establish which the most and least significant aesthetics were for graph usability. The results of the study were that maximising incident edge angles and orthogonality did not lead to better performance, preserving symmetry and minimising line bends were somewhat important, and minimising edge crossings was the most significant influencing factor for performance [Purchase, 1997].

Metro maps, however–particularly those representing non-physical data such as in the previous section–do not share all the structural principles of the wider set of directed graphs. Consequentially, Stott et al. developed a set of criteria for metro maps specifically, including criteria for the labelling of stations. The criteria are as follows [Stott et al., 2011]:

- **Angular Resolution Criterion:** Maximise the angle between incident edges at each node. This criterion is also the fifth principle in [Purchase et al., 1997].
- **Edge Length Criterion:** All edges on the map should be approximately equal in length.
- **Balanced Edge Length Criterion:** The length of edges incident to a given node should be approximately equal.
- **Edge Crossings Criterion:** Crossings should be minimised. This criterion is also the second principle in [Purchase et al., 1997], which was identified as the most important.
- **Line Straightness Criterion:** Edges on the same metro line should be collinear where possible. That is, they should form a 180° line through every node that the line passes through. This criterion relates closely to the first principle (line bends) in [Purchase et al., 1997].
- **Octilinearity Criterion:** Edges should be drawn at multiples of 45°. This criterion is an extension to the fourth principle in [Purchase et al., 1997], as Tamassia's [1987] orthogonality is analogous to rectilinearity.

Evaluating Figure 1.5 according to the list above, we observe that despite only containing 16 stations, the map unnecessarily violates five of the six Stott et al. criteria; all apart from edge crossings.

Therefore, to improve the usability of the metro maps drawn in [Shahaf and Guestrin, 2010, Shahaf et al., 2012*b*,*a*, 2013], it would be advisable to attempt to satisfy more of the aesthetic principles for graphs and metro maps described in [Purchase et al., 1997, Stott et al., 2011].

1.4 Towards Newsfeed Visualisation

The fact that news articles form a fairly narrow class of document is an advantage from a visualisation design perspective, due to the common elements they share. Articles published by commercial news producers typically contain:

- A headline;
- A meta-description, or *subhead*;
- A publish date;
- One or more categories to which the article belongs.

These attributes are useful for visualisation, since creating a spatial representation from text requires documents to be represented as vectors in high-dimensional feature space [Wise et al., 1995], and the presence of existing attributes makes articles more inherently comparable than their unstructured contents would be.

There is also a well-known existing standard for publishing links to articles with their metadata for use by other applications; RSS.

1.4.1 Content Retrieval

The de-facto web format for feed publishing is RSS (Rich Site Summary, or Really Simple Syndication.) The rise of the internet as a news platform has lead to many readers finding the most efficient method of reading news articles is to subscribe to various topic-specific newsfeeds and read what is automatically collated by their computers [Wang et al., 2006].

Although RSS–which is a subset of XML–is standardised[2], the practice of feed categorisation is not, meaning the granularity of topics which can be subscribed to is dependent on the publisher. This issue was addressed by Liu, Han, Noro and Tokuda [2009], with the design of a system which could essentially split or join existing RSS feeds to synthesise new ones based on user-specified keywords and queries.

Despite its shortcomings, RSS remains the most universal option for accessing feed content from a wide variety of news producers [O'Shea and Levene, 2011].

[2] `http://cyber.harvard.edu/rss/rss.html`

1.4.2 Keyword Extraction

Extracting relevant keywords from documents is not a new domain of research. Various methods have been presented, the most well-known being the intuitively logical tf-idf (term frequency, inverse document frequency) [Salton and Buckley, 1988] which ranks the significance of a term t in a document d which belongs to a corpus C as follows:

$$\text{tf-idf}(t) = \frac{Occurrences(t,d)}{WordCount(d)} \times log_e\left(\frac{|C|}{|\{c \in C \mid t \in c\}|}\right) \tag{1.6}$$

Tf-idf will extract the most unique keywords from a document within a corpus, because it penalises words which are common to many documents. However, in the context of a corpus of news articles, this uniqueness can lead to significant topic keywords being ignored because they appear with such frequency.

Bun and Ishizuka [2002] found that for news archive keyword extraction, a better alternative to tf-idf is tf-pdf (term frequency, proportional document frequency) as it is not biased against frequently repeated keywords.

Using tf-pdf, articles are modelled as belonging to one of a finite number of sources or *channels* within a corpus. The weighting of a term from an article within a channel is in this case linearly proportional to its frequency in the channel and exponentially proportional to the number documents in the channel where it occurs.

A term's total weighting is the sum of its weightings across all channels, as can be seen in Equation 1.7 [Bun and Ishizuka, 2002], where:

- D = The number of channels in the corpus;
- K_c = The total number of terms in channel c;
- F_{tc} = Frequency of term t in channel c;
- n_{tc} = The number of articles in channel c where term t occurs;
- N_c = The total number of articles in channel c.

$$\text{tf-pdf}(t, D, K) = \sum_{c=1}^{c=D} \frac{F_{tc}}{\sqrt{\sum_{k=1}^{k=K_c} {F_{kc}}^2}} \times \exp\left(\frac{n_{tc}}{Nc}\right) \tag{1.7}$$

An approach derived from energy levels in quantum systems was proposed in Carpena et al. [2009], where keywords were extracted based on their spatial distributions within a single text. The theory behind the approach is that typically, keywords occurrences are distributed in significant frequency clusters throughout a document, whereas non-relevant words are distributed with uniform frequency (see Figure 1.7). This technique allows relevant keywords to be distinguished from non-relevant common words with similar total frequencies without the use of a background corpus for comparison.

Several important observations have been made regarding extracting keywords from news articles specifically. Firstly, that important phrases in text are likely to be references to people, places and other named entities [Teitler et al., 2008]. Libraries such as the Stanford

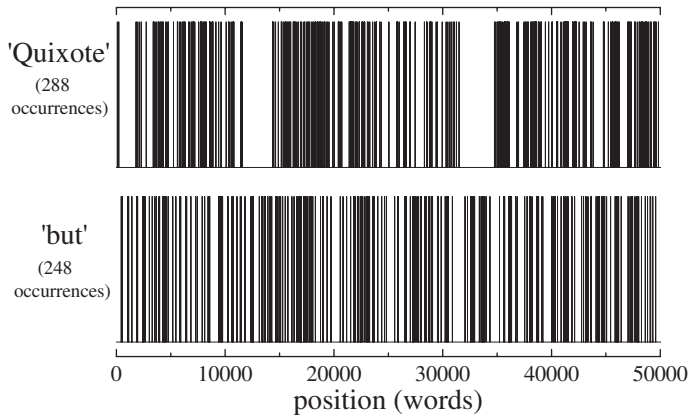


Figure 1.7: Frequency spectra for 'Quixote' and 'but' in the first 50,000 words of *Don Quixote* [Carpena et al., 2009].

Named Entity Recognizer (NER) [Finkel and Manning, 2009] exist to extract these from text. Secondly, that while 30% of an article's keywords are inferred and cannot be found within the text without intelligent input, 60% are present in the article's title and first few sentences [Lin and Hovy, 1997], since important facts are generally stated as part of an article's *above the fold* content.

1.5 Evaluation Methods

Shahaf et al. [2012*b*] evaluated their system both for accuracy and with a user study, though as previously discussed they did not evaluate the aesthetic properties of the generated maps. The accuracy evaluation tested whether the system included the most 'important' (as decided by academic experts) documents in the map.

The user study focused on the strength of the results returned by specific queries, where output was transformed into a structureless list in order for the study to be double-blind against the other methods. The evaluation was performed between-subjects, so background knowledge had to be controlled for. Output was compared with that from Google News and a TDT (Topic Detection and Tracking) method presented in [Nallapati, 2003]. This approach to evaluation would not be appropriate for the proposed system, as the authors were actually evaluating the performance of the system in selecting documents based on a query, rather than visualising the chosen documents on a metro map. In contrast, the usability of the visualisations produced is precisely the aspect of our process which would need to be evaluated.

The evaluation of TIARA [Liu, Zhou, Pan, Qian, Cai and Lian, 2009] is more relevant to the proposed system, as it directly evaluated the visualisation system against a baseline application which did not share any of its advanced features, although the two were tailored for the same task; email analysis. A series of questions were asked of participants, who used either TIARA or the baseline system to answer. The response time and accuracy of the participants was recorded, as well as their levels of satisfaction after completing the task.

Although Stott et al. [2011] could have used their own layout criteria to quantitatively evaluate the metro maps produced by their optimisation function, this was not how they chose to evaluate their system. Instead, the authors conducted a study where they evaluated participants based on their task performance during a route-finding problem. They also compared preferences between groups, where for each task, each of the three groups rotated between using an automatically-drawn metro map, an undistorted map, or the official published map for the city's metro.

Route-finding was a logical choice of task for the Stott et al. metro maps, as the maps produced were variants on the metro maps of existing cities rather than metaphoric representations of other data. In contrast, there is no such simple atomic task which is representative of typical news reading behaviour. The difficulties of choosing a task by which the proposed system could be evaluated are discussed in detail in Chapter 5.

1.6 Summary

To recap, this review began with an exploration of the problem of news overload using the findings of The Associated Press and Context-Based Research Group [2008], who conducted a field study into the news consumption habits of young people. The findings of the study were discussed in the context of four dimensions of information overload [Ho and Tang, 2001, Bergamaschi et al., 2010], as was the relationship between combating overload and supporting sensemaking; the cognitive process this project aims to support.

Information visualisation was identified as one common approach to both supporting sensemaking and reducing information overload, so as a result of this, we provided a high level overview of various methods for visualising text-based documents. Visualisations for specific classes of document [Goyal et al., 2013, Havre et al., 2002, Liu, Zhou, Pan, Qian, Cai and Lian, 2009, Singh et al., 2015, Wang et al., 2006] were presented and compared, in addition to a more in-depth exploration of visual metaphors–in particular timelines–and schematic map visualisations. This section also featured the introduction of the metro map in its original usage context, before moving on to its use as a metaphor for visualisation in various domains.

Centrally, the work of Shahaf et al. [Shahaf and Guestrin, 2010, Shahaf et al., 2012*b*,*a*, 2013] which is particularly relevant to the aims and objectives of this project was discussed in detail, with an explanation of the desirable properties of metro map content (*coherence*, *coverage* and *complexity*). Our main critiques of this body of work were of the fixed background corpus used, and the physical layouts of the drawn maps when evaluated against the criteria defined in [Purchase et al., 1997, Stott et al., 2011].

Next followed a practical overview of methods for transforming news articles into textual entities with comparative properties, including RSS feed mining, entity recognition, and keyword extraction. Lastly, approaches to experimental design and evaluation by several of the aforementioned studies were discussed.

The literature, techniques and terminology discussed in this review will be taken forward into the subsequent chapters, as this background material was highly influential in the design and implementation of the system.

Chapter 2
Requirements

This chapter discusses key choices which were made in the scoping and specification of the system and provides a high-level guide to its architectural design. The full requirements specification can be found in Appendices A.1 and A.2.

2.1 Project Scope

For the purpose of this dissertation and its timescale, the goal of the project was specified as the development of a system which generates metro maps based on the content of user-specified RSS feeds. The subject of news fatigue and its possible remedies span multiple domains from data science to cognitive psychology to journalism and content publishing, so while there are many possible techniques in the feasible scope of the project, the practical scope required significant narrowing.

No focus was given to the content of the RSS feeds themselves, as the standard is sufficiently specified in order for its attributes to be usable without the need to understand their content. Although it is a recognised problem that there is no standardisation for the granularity of RSS feed categories [Liu, Han, Noro and Tokuda, 2009], the only effect varying this would have on the system would be the granularity of the output visualisation. Likewise, RSS feed discovery and recommendation, while clearly a potential extension to this work from the perspective of improving usability, was not considered.

In terms of the dimensions of information overload [Ho and Tang, 2001] discussed in the previous chapter, while a degree of attention was given to addressing all four dimensions, the focus was on information quality–in particular, contextual quality–and fragmentation. Information format, albeit an important facet of overload in general, does not vary significantly between major news publishers. Therefore, while this project may contribute a new unified format for displaying a collection of articles, changing the format or textual content of the articles themselves is not within its scope; any changes which do occur are incidental.

The focus of this work is ultimately the implementation of an end-to-end process for transforming news feeds into metro map schematics; that is to say, few of the techniques themselves are new. The area of interest is the transformation of data from articles to points in physical space on a transit map, and how varying the functions which compose the transformation affect the resultant metro maps.

2.2 Requirements Gathering

This project was primarily research-based, so while typical software engineering practices were followed through the creation of a requirements specification for organisational purposes, we did not undertake a formal requirements gathering with involving potential users.

Instead, the system's requirements were derived from the background work discussed in the previous chapter; features of the existing news visualisation system [Shahaf et al., 2012*b*], the transit map aesthetic principles formulated in [Stott, 2008, Stott et al., 2011], and Shneiderman's [1996] infovis task taxonomy. Underpinning the technical requirements are the high-level recommendations for news producers specified by The Associated Press and Context-Based Research Group [2008] and Nordenson [2008].

2.3 Prioritisation

Requirements were assigned priorities using the MoSCoW technique, since the size of the project was not large enough to warrant a more formal system. MoSCoW assigns requirements to one of four categories [Waters, 2009];

- **M**ust have: Essential features required for the project to be useful.
- **S**hould have: High value but non-critical features.
- **C**ould have: Desirable features which could be moved out of scope if necessary.
- **W**on't have: Features which were requested but were not included.

As our requirements gathering process did not include input from potential users, there were no requirements with a *won't have* modifier.

2.3.1 Categorisation

The operation of the system forms a pipeline of four components through which data is transformed, with each component comprising some distinct functionality which can be designed, implemented and if necessary modified, in isolation. The components are as follows:

1. *Article Retrieval*: The process of parsing an RSS feed and downloading content from the articles it syndicates.
2. *Keyword Extraction*: The language processing component, wherein articles are tokenised and their significant keywords are extracted.
3. *Graph Building*: The transformation of a collection of articles and their associated keywords into a graph structure, by selecting keywords which best represent the entire corpus. The graph has no physical layout during this stage of the pipeline.
4. *Map Drawing*: The generation of a visual representation of the graph structure in the form of a metro map, which the user will interact with.

In addition to the four stages of the pipeline, the system requires an ancillary storage component, to allow processed corpora and their graphs to be imported and exported. In the specification, all functional requirements were grouped according to one of the four stages, to assist in the implementation planning and testing processes.

2.4 Architectural Specification

Figure 2.1 outlines the decomposition of the four pipeline components into their high-level subtasks, with descriptions of the data transformations which occur at each stage.

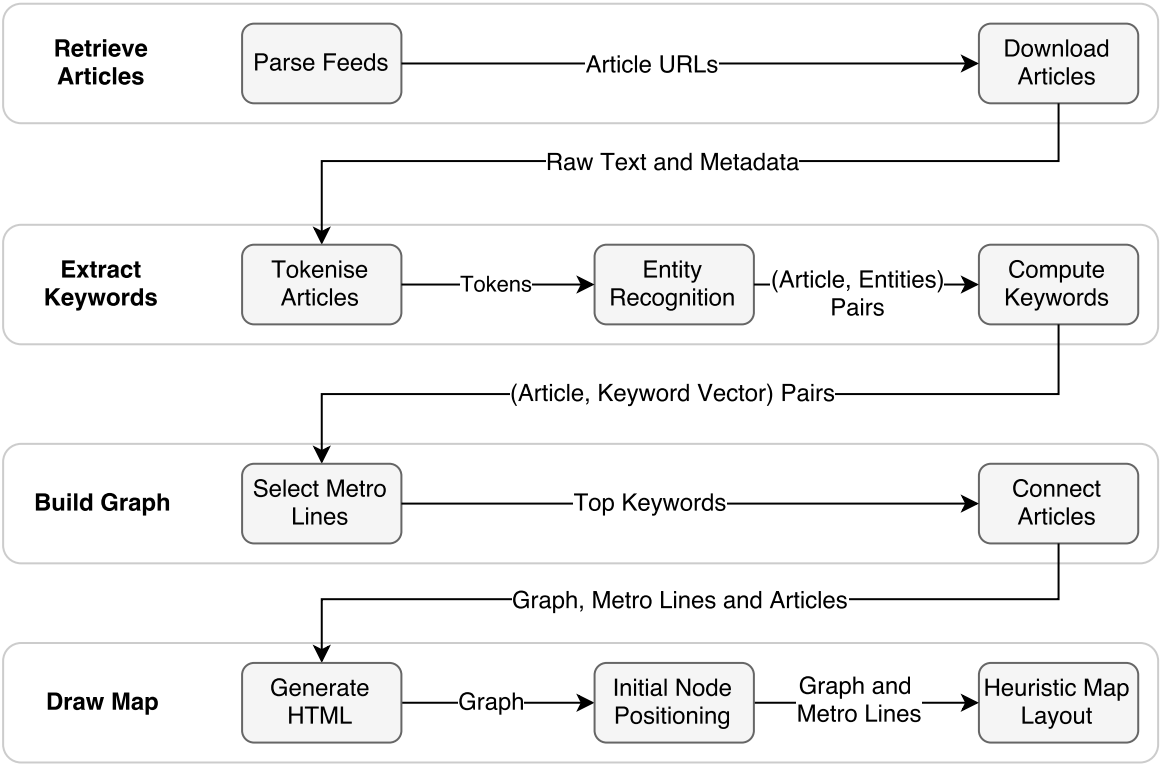


Figure 2.1: A conceptual model of data flow between components of the system.

Not included in Figure 2.1 is the boundary between the *Build Graph* and *Draw Map* stages of the pipeline, which marks the transition from the run-time Python where article data is extracted and processed, to JavaScript which only runs once the visualisation is opened in the browser; this comprises the entirety of the *Draw Map* stage.

The boundary takes the form of an intermediate layer between *Build Graph* and *Draw Map*, where the topological map data and any article metadata required for the visualisation are serialised to JSON[1] and written to a template HTML file, which then contains the metro map in a standalone, shareable cross-platform format.

[1] JavaScript Object Notation; `http://www.json.org`

2.5 Discussion

Central to the importance of the project is the knowledge that news consumers do not subscribe to single RSS feeds; they specifically seek out software which aggregates multiple feeds for convenience [Wang et al., 2006]. Consequentially, F1.1 specifies that the system must accept multiple RSS feeds. However, news preferences are diverse, which gives rise to the potential case where a user specifies multiple feeds which do not share any significant keywords, and articles from one feed are excluded from the visualisation due to their lack of connectivity with the others.

Initially, we considered a requirement for the connectivity [Shahaf et al., 2012*b*] of every metro line to be greater than one; that is, no line should be included in the visualisation unless it intersects with another. However, in the case described above, mutually exclusive RSS feeds could lead to 'orphaned' lines which still contain useful content but do not share contextual links to the main body of the map. The stories on these lines should not be ignored by the system, as they most likely would not be ignored by a user browsing with a traditional RSS reader. Our argument is that the lack of connectivity of an article can itself be viewed as metadata on that article (in the sense that it is a corpus outliter), and is not something to be penalised. Although a map which only contains orphaned lines is indicative of poor line selection, or–though this is less likely–a feed containing a set of completely unrelated articles, a small number of orphaned lines for any given news corpus is to be expected. This highlights a fundamental assumption of this work; that in order to be of use, RSS feeds should contain explicitly related content.

A second issue of deliberation was the specification that stations on the generated maps should not be labelled (F4.7). The use of node labels is logical for a search task, where locating a particular entity or route on a map is the goal. However, when the goal is gaining an insight into the structure of the collection being visualised, labelling each station with the title of the article it represents would only reintroduce the information overload we are attempting to mitigate. The *overview* in the "Overview first" clause of the information seeking mantra [Shneiderman, 1996] is by definition the highest level of abstraction available on each data point, and while in a typical RSS reader this may well be the title, in our system this will not be the case. The context of the articles, that is, the metro lines themselves are what provide the overview, and the titles instead fall under "details-on-demand."

One requirement which was left deliberately vague was F4.8; where possible, maps should comply with Stott's [2008] aesthetic criteria for metro map layouts. The lack of strictness and specificity in this requirement was due to the non-spatial nature of the data being represented. While it is likely that there is an optimal layout for metro maps schematising real transit networks due to the geographic positioning of stations on those networks, there is no 'sensible' underlying topology to metro maps of news to serve as a starting point. It was unknown at this stage whether we would be able to enforce strict layout criteria without significant pruning of the data.

Chapter 3
Design and Implementation

During the requirements gathering process, four distinct components of the system were identified which form a pipeline of execution; *Article Retrieval*, *Keyword Extraction*, *Graph Building* and *Map Drawing*. Crucially, all key areas of functionality within components are decoupled by design, allowing both for flexible extensibility and for alternative implementations to be tested directly against each other without requiring changes to the other modules.

This chapter decomposes each of the four components in order, and provides a detailed overview of their design and implementation, including a discussion of significant challenges and successes which arose during development.

3.1 Code Reuse

The system was developed in Python 2.7 and JavaScript 1.7 using various open-source libraries and APIs. The most notable are detailed below and discussed in context in the following sections.

- **FeedParser**[1]: A Python module for downloading and parsing RSS feeds.
- **Goose Extractor**[2] A Python library for downloading and extracting cleaned text and metadata from online articles.
- **lxml**[3]: A Python library for generating, parsing and manipulating XML and HTML.
- **NLTK (The Natural Language Toolkit)**[4]: A Python library for natural language processing and analysis.
- **Google Knowledge Graph Search**[5]: The API for Google's knowledge base, which returns structured semantic search results.
- **D3.js**[6] (v2): A JavaScript library for creating and manipulating interactive data-driven web visualisations.

[1] `http://pythonhosted.org/feedparser`
[2] `https://pypi.python.org/pypi/goose-extractor/`
[3] `http://lxml.de`
[4] `http://www.nltk.org`
[5] `https://developers.google.com/knowledge-graph`
[6] `http://d3js.org`

3.2 Article Retrieval

The first stage of article retrieval is the parsing of RSS feeds, in order for article URLs and metadata to be extracted. This component of the system is substantially smaller than the remaining three, as it is simply a preprocessing task.

The Python library FeedParser was used for parsing, as at the time of writing it provided the best support for RSS 0.9x, 1.0 and 2.0. The parsing process extracts a link, channel name, and the parsed publish date from every article in the feed, but it will also attempt to extract the author name if the `author` attribute is found.

Once the feed data has been extracted, it is used to construct an instance of `ArticleCollection`, which acts a wrapper around the contents of one or more feeds and provides the mechanism necessary to perform corpus-wide queries. The `Article` class encapsulates the functions for computing article-specific terms such as term-frequency for tf-idf, and the term-weighting component of tf-pdf.

3.3 Keyword Extraction

The keyword extraction stage begins with the process of tokenising the body text of every article. Tokenisation here has three substages, all of which were implemented using NLTK (The Natural Language Toolkit for Python), and which are as follows (see Figure 3.1):

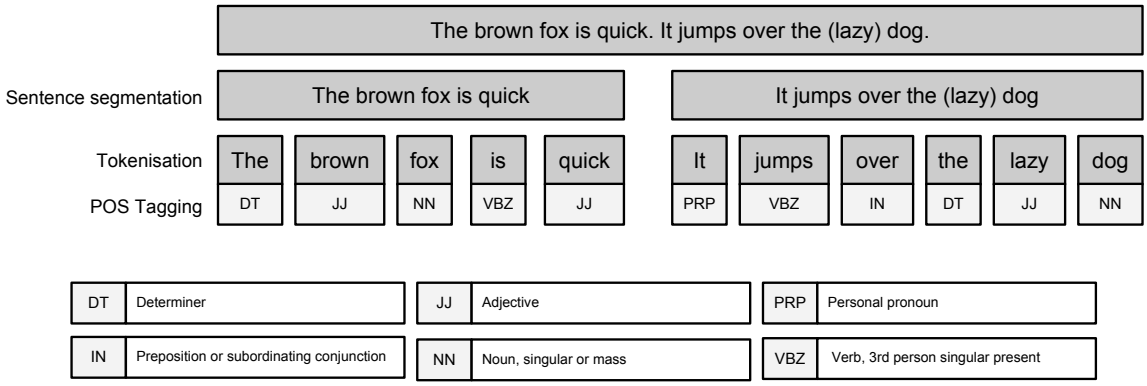


Figure 3.1: Stepping through the tokenisation process

1. **Sentence segmentation:** Split the text into a list of sentences and remove sentence punctuation. This process is important, to prevent phrases being mistakenly identified across clause boundaries. It is also a non-trivial problem due to the ambiguity of the full-stop in English language texts [Palmer, 2000].
2. **Tokenisation:** Split each sentence into a list of individual words and remove both whitespace and any remaining clause punctuation.

3. **Part-of-Speech (POS) tagging:** Categorise each token according to its lexical class, e.g. adjective (*JJ* in NLTK). This is more of an extension to the tokenisation process than strictly a part of it, but it is still necessary for subsequent processing.

Once the raw article bodies and titles have been tokenised and tagged, the next subproblem is identifying words and phrases of importance within articles. These keywords will form our set of candidate metro lines in the next stage of the pipeline.

3.3.1 Named Entity Recognition

The second half of the keyword extraction process is exclusively concerned with named entities within articles. This is because, intuitively, the names of people, places, events, companies and other *things* form a set of strong candidate keywords.

Keyword extraction from named entities has been empirically shown to outperform other methods for extraction such as using only noun phrases [Sayyadi et al., 2009]. Additionally, restricting the candidates to entities alone reduces the search space by an order of magnitude when using frequency-based methods for keyword extraction, and bypasses the need for other natural language processing tasks such as stop-word removal and lemmatisation.

After tokens have been POS tagged, named-entity chunking can be performed, again using NLTK. This process groups tokens into contiguous and non-overlapping *chunks*, where each chunk is a named entity; typically a proper noun or some other noun phrase. It is possible for chunks to contain other chunks (consider the chunk 'Bank of England', which contains the chunk 'England'), but this is typically undesirable for entity recognition, where we desire specificity. Consequentially, after chunking the tokens, we flatten the chunk structure so chunks cannot be any further decomposed.

The result of this process is a list of named-entity chunks, each of which will be an eventual candidate for becoming a metro line. However, the chunks require some further processing before we can determine whether or not they are significant keywords within the source text.

3.3.1.1 Substring Matching

It is common stylistic practice in all forms of journalism to refer to the subjects of articles by their surnames. However, to avoid any confusion, the full names of those mentioned will often appear in the title or first few paragraphs of the article. If keyword strength is determined by frequency, then regardless of whether we use tf-idf or tf-pdf, every occurrence of an entity must refer to that entity by the same name; preferably the most specific, which is typically the longest.

During this stage of disambiguation, we only consider entities which are mentioned twice or more in the source text as candidates. This is because the chunking process can occasionally produce false positives by combining unrelated adjectives with valid chunks, but the likelihood of the same false positive being produced multiple times within the same article is much lower.

To begin the process of substring matching, let P denote the set of entities with strictly more than one occurrence in the source article; S:

$$P = \{e \mid \text{occurences}(e, S) > 1\}$$

$\forall (e_1, e_2) \in \{P \times P\}$ if e_1 is a substring of e_2, we call e_1 an *alias* of e_2. Due to the substring restriction, aliasing is not commutative. Let A be the set of alias pairs (e_1, e_2) in S;

$$A = \{(a, e) \mid a \text{ is an alias of } e\}$$

We require the first term of every pair in A to be unique, but there is no such constraint for the second term (e); this allows multiple aliases to map to the same entity. For example,

$$\{(\text{'Zuckerberg'}, \text{'Mark Zuckerberg'}), (\text{'Zuck'}, \text{'Mark Zuckerberg'})\}$$

would be unambiguous and therefore valid, but

$$\{(\text{'Mark'}, \text{'Mark Zuckerberg'}), (\text{'Mark'}, \text{'Zuckerberg'})\}$$

would not be allowed. Let U be the set of unambiguous alias-entity pairs in S:

$$U = \{(a, e) \mid (a, e) \in A \cap \forall (e_1, e_2) \in A,\ a = e_1 \implies e = e_2\}$$

We have a reasonable degree of confidence that for every alias-entity pair $(a, e) \in U$, occurrences of a in the source text can be replaced by e, making e a stronger candidate for keyword detection. Pairs in $(a, e) \in A \setminus U$, that is, aliases which could map to multiple entities in the source, are disregarded at this stage and left unchanged. Algorithm 1 illustrates how, given a list of entities, the unambiguous pairs can be found deterministically.

Algorithm 1: Finding unambiguous alias-entity pairs

```
   Data: names: a list of recognised entities
   Result: U: the set of unambiguous pairs in names
 1 U ← {};
 2 foreach e1, e2 ∈ (names × names) do
 3 |   if len(e1) > len(e2) then
 4 |   |   swap(e1, e2) ;
 5 |   end
 6 |   if e1 ∈ e2 then                          /* If e1 is a substring of e2 */
 7 |   |   if e1 ∉ U then
 8 |   |   |   U[e1] ← e2 ;                    /* (e1, e2) are a candidate pair */
 9 |   |   else
10 |   |   |   delete U[e1] ;                  /* e1 is now ambiguous, so remove it */
11 |   |   end
12 |   end
13 end
```

3.3.1.2 Entity Disambiguation with Knowledge Graph

The process described in the previous section is a simplistic approach which only addresses the matching of partial name mentions to full mentions. In comparison, *entity disambiguation* refers to the process of determining the identity of entities in a body of text. Performing this process on the sentence 'The UK has voted to leave the EU,' should identify 'UK' as 'The United Kingdom' and 'EU' as 'The European Union'. As advanced methods for entity disambiguation are both complex and computationally expensive, we did not implement a formal method for this.

Instead, the list of recognised entities are queried against Google's publicly accessible Knowledge Graph API, which returns a list of potential results as Schema.org[7] types. Knowledge Graph is the service which replaced Freebase in 2015, and currently contains over 70 billion facts [@jeffjarvis Twitter account, 2016].

Dredze et al. [2010] identify three key challenges in entity linking using knowledge bases; name variations, ambiguity, and absence. Absence describes the lack of a corresponding entry for the entity in the knowledge base, which is not an easily tackled problem. Ambiguity is a consequence of the polysemy of many names and acronyms and requires disambiguation to be performed using more contextual methods.

Both of these problems are left unsolved in the design and implementation of the system due to scoping constraints, although absence is less of a concern in current events news texts. Naïve substring matching however, combined with the use of Knowledge Graph and some empirical parametric estimation, are enough to disambiguate name variations of all forms (partial matches, abbreviations, and acronyms) to a sufficient degree and with surprising accuracy.

The key to using Knowledge Graph for disambiguation lies in its most vaguely defined return value. Each result has a score attributed to it by Knowledge Graph, which is an indicator the strength of the match between the entity and the original query. Results are sorted by descending score; the higher the `resultScore`, the better the match. Although there is no defined upper limit for this value and no official documentation on how the score is derived, comparing the scores of the top two results for a query can provide a measure of certainty, for all but particularly esoteric or unknown entities.

We specify a threshold $\frac{1}{t}$, which is roughly proportional to the likelihood of accepting a false positive match. Then, if dividing the score of the first result by the score of the second yields a number greater than $\frac{1}{t}$, the match is accepted. Provisionally we set $t = 0.5$; a discussion of how this value was derived is provided in the next section.

Using a knowledge base for disambiguation also inadvertently solves another problem we encountered while tokenising and parsing articles, this time as a result the expositional style of journalistic writing. While keywords are typically nouns or noun phrases, they can also appear in the form of denominal adjectives. These adjectives are derived from nouns; e.g. 'French' implies 'France' might be a keyword. Denominal adjectives are not amenable to traditional stemming or lemmatising, but querying Knowledge Graph for 'French' returns a top match of 'France' with a `resultScore` of 432.42807; more than four times larger than the next result.

[7] http://schema.org is an online hierarchy of types managed by W3C (The World Wide Web Consortium)

Given a list of entities E, and top two results of a Knowledge Graph query for each $e \in E$; $R_e(1)$ and $R_e(2)$ with scores $S_e(1)$ and $S_e(2)$ respectively, our aim is to return a set of zero or more pairs mapping entities in E to their disambiguated forms;

$$K = \left\{ (e, R_e(1)) \mid \frac{S_e(1)}{S_e(2)} > \frac{1}{t} \right\}$$

Algorithm 2 describes this process. For higher values of t, the likelihood of accepting a false positive increases, and for lower values, the likelihood of accepting a false negative increases. For corpora which do not contain references to public figures and place names, choice of t should be empirically tuned against the prevalence of the expected entities.

Algorithm 2: Entity disambiguation with Knowledge Graph

```
Data: names: a list of recognised entities
      t: the acceptance threshold for results, default = 0.5
Result: K: a set of disambiguation mappings for elements in names
K ← {};
T ← 1 ÷ t;
foreach e ∈ names do
    results ← KnowledgeGraphResults(e);
    if len(results) > 1 then
        if results[0].score ÷ results[1].score > T then
            K[e] = results[0].name;
        end
    end
end
```

It is unclear what should become of queries that Knowledge Graph only returns a single result for, but in this implementation they are ignored. Similarly, although Algorithm 2 terminates with a set of unambiguous mappings, there is still the potential for the system to identify mappings such as:

$$K = \{(\text{'Zuck'}, \text{'Zuckerberg'}), (\text{'Zuckerberg'}, \text{'Mark Zuckerberg'})\}$$

which simplifies to:

$$K = \{(\text{'Zuck'}, \text{'Mark Zuckerberg'}), (\text{'Zuckerberg'}, \text{'Mark Zuckerberg'})\}$$

A simple process by which the pairs in K can be reduced as follows:

$$Reduce(\{(e, R_e), (f, R_f)\}) = \{(e, R_f), (f, R_f)\} \text{ if } R_e = f$$

This would require careful removal of any circular aliases; removing cases where

$$K = \{(e, R_e), (R_e, e)\}$$

Pairs in K could then be iterated over, applying this reduction rule until there are no remaining aliases which are themselves aliased to.

3.3.1.3 Determining Optimal Values for t

Empirically, we found the best values for t were in the range $0.35 < t < 0.65$, meaning the system accepts top results which are at least 1.5x-3x higher than the next best candidate. To determine this, we logged all the disambiguation pairs found in 20 articles from the BBC Politics RSS feed, letting t range over $\{0.3, 0.5, 0.7, 0.9\}$. These pairs were then manually classified as either true or false positives, where a true positive indicates a correctly identified entity, and a false positive is either an incorrectly disambiguated entity or a non-entity which was mistakenly recognised. Figure 3.2 shows the results of this investigation.

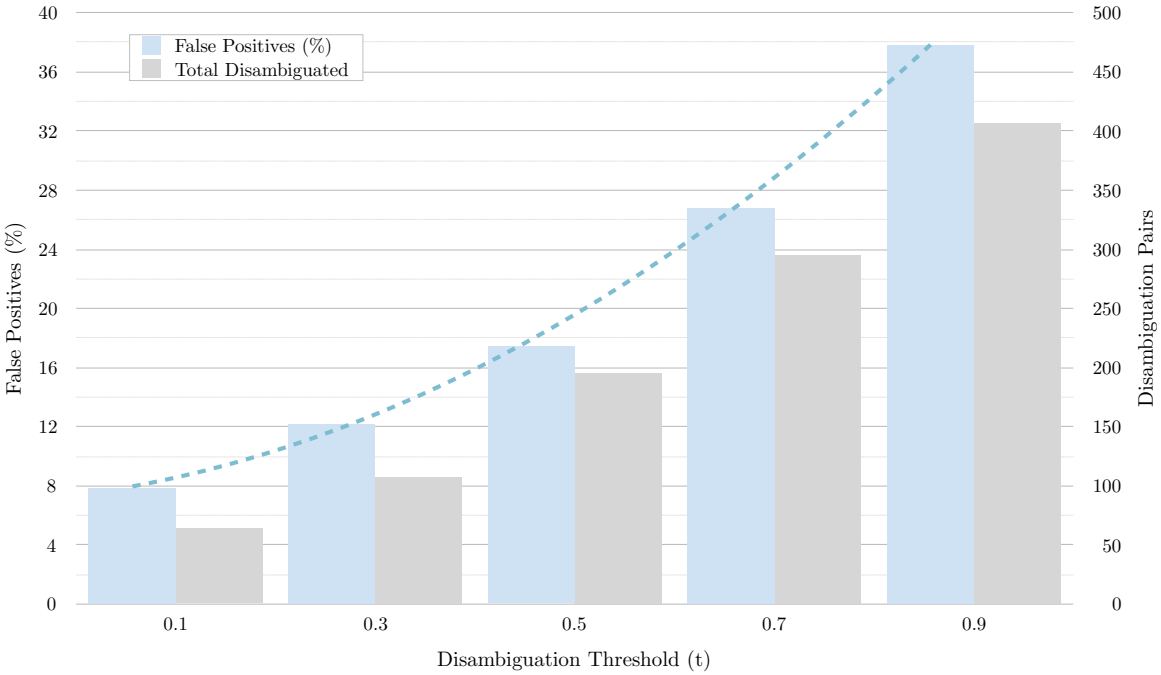


Figure 3.2: Disambiguation threshold against pairs found and false positives (%)

Although a reduction can be seen in false positives for smaller values of t, the trend-line illustrates that this is a game of diminishing returns; reducing t from 0.3 to 0.1 only reduces the false positives by 4.34%, but it leads to 43 fewer pairs being identified[8].

There is a clear trade-off between the accuracy of the disambiguation and the number of false negatives which are discarded, but the cost of false positives in this case is less than the cost of false negatives. While a false positive could result in unrelated entities appearing as keywords for certain articles, the likelihood of the same false positive appearing with high frequency in enough articles to result in an erroneous metro line is incredibly low. In contrast, the cost of disregarding a false negative could result in articles being left off certain metro lines altogether. It is for this reason that we do not simply choose the value of t which yields the minimum ratio of false positives.

[8]Since it is both extensive and tangential to the focus of the project, raw data for this table can be found at `http://bit.ly/DisambiguationThresholding`.

3.3.2 Choosing a Term-Weighting Metric for Keyword Ranking

Given a set of disambiguated entities for every article, the union of which forms a set of candidates keywords or *metro lines* for the collection, we must now determine which keywords are the most relevant. To do this, the two term-weighting methods described in Chapter 1 will be revisited; tf-idf [Sparck Jones, 1972] and tf-pdf [Bun and Ishizuka, 2002]. Both were implemented in the system so they could be compared.

From an implementation perspective, the main difference between these two algorithms are the level at which they operate. Tf-idf ranks keywords on a per-article basis, returning a vector of an article's keywords and their corresponding score against the background corpus. In contrast, tf-pdf is specified at a corpus level and only returns a single metric for a given word in a corpus; the sum of its significance across the whole corpus. When using tf-pdf, corpora are decomposed into one or more *channels*, which contain articles. In our system, channels can be conveniently defined as RSS feeds from different publishers.

Bun and Ishizuka argue that tf-pdf is a more suitable metric for news corpora, because the algorithm discriminates between articles which originated from the same channel and articles in different channels, allowing keywords which are highly (and uniformly) frequent in one channel to be identified as significant in documents which originated from a different channel. Similarly to tf-idf, this process can be used to produce a vector of the highest ranked keywords within a corpus. Tf-idf however requires an additional later level of selection to transform the keyword vectors of each individual article into one global keyword vector of candidate metro lines for the corpus.

The actual scores attributed to keywords by either algorithm are not of direct importance, since it is only relative scores which are used to construct the keyword vectors. Regardless of which algorithm is used, the advantage to restricting the set of candidates to named entities can be quantified.

Figure 3.3 shows the number of tokens against the percentage of extracted entities for 40 articles from the BBC's Politics RSS Feed.[9] The interquartile range shows 50% of articles had entities comprising 5.4%-8.2% of their tokens after stop-word removal. With the mean equal to 6.65%, the implications of this are a search space which can be reduced by a factor of more than ten. Although performance optimisation was not specified in the aims of the project, gains of this nature are still significant.

Once articles have been reduced to vectors of entities, given optional user-specified lists of keywords to either *include* or *ignore*, the system can penalise or boost the scores of articles which mention these topics, which in the case of the *include* list will almost guarantee the keyword is selected as a metro line in the map, as long as it is present in the keyword vectors of at least two articles.

The function of the *ignore* set is not to blacklist articles which mention certain topics from the map, as it will have no effect on such articles being placed on other metro lines. Instead, it performs domain specific stop-word removal for metro line candidates, preventing non-useful keywords with inevitably high scores (such as 'United Kingdom' for a metro map based on a UK news corpus) from being selected.

[9] http://feeds.bbci.co.uk/news/politics/rss.xml (Accessed: 27/02/2017, Full data in Appendix B.1)

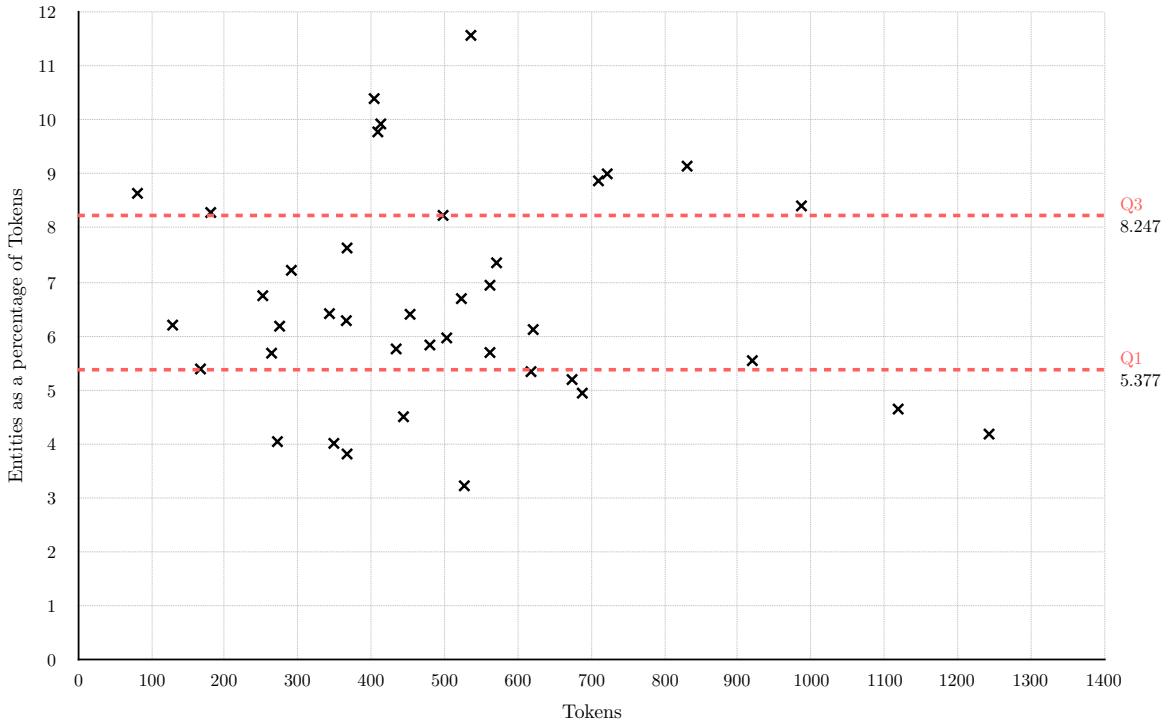


Figure 3.3: Entities as a percentage of Tokens across 40 BBC Politics articles.

During implementation, it was found to be useful to automatically append the names of every news publisher (e.g. 'The Guardian') and author in the feeds to the *ignore* set, as well as the names of the feeds themselves. This forces the system to select more specific candidates, even though their scores may be significantly lower.

While implementing the *ignore* functionality the differing results of using tf-idf and tf-pdf on the same corpus became obvious. Because tf-idf penalises words which appear with low but constant frequency across a number of documents, keywords we intuitively would wish to ignore such as 'United Kingdom' are not scored as highly as they would be with tf-pdf.

This is in line with the findings of Bun and Ishizuka; tf-pdf is more useful for detecting low-frequency keywords which are consistent across many documents. However, this behaviour is exactly the opposite of what we desire from the metro lines produced by the system; those low-frequency keywords are most likely already known or could be easily inferred by a user reading the map.

We therefore recommend tf-idf be given preference over tf-pdf in any implementations of metro map systems for news, where the names of more specific themes or topics are being sought. The exception to this is when using a corpus constructed from RSS feeds which share very few topic themes. If this is the case, tf-idf will struggle to extract any topics common enough to form metro lines from, resulting in an overly sparse map. The higher-level topics which are less severely penalised by tf-pdf will provide the most cohesive overview in these instances.

3.4 Graph Building

The key challenge of the graph building process is generalising the characteristics of a good overview, from both news consumer and metro map user perspectives. It is at this point that a significant difference emerges between this system and the work of Shahaf et al. [2012*a*,*b*, 2013].

Shahaf et al. generated maps in response to a query (such as the name of an event or time) period, meaning the entire map would then be oriented around that query. In contrast, in our system the only query is an implied one; "What's going on today?" which offers no starting point for choosing metro lines. Instead of a query-based search problem, we have a multi-document summarisation problem. While they were choosing a set of lines to fit a given query, we are concerned with choosing a set of lines which best cover an entire corpus.

3.4.1 Extending the Definition of Coverage to Paths

Our starting point in this section is a corpus of articles and their associated keyword vectors, which form a metro map by Shahaf et al.'s definition, but one with too many edges and metro lines to be readable in practice, as Figure 3.4 illustrates. Therefore, in order to selectively prune the number of paths to some fixed upper bound, a metric is required to calculate the relative importance of paths within a corpus.

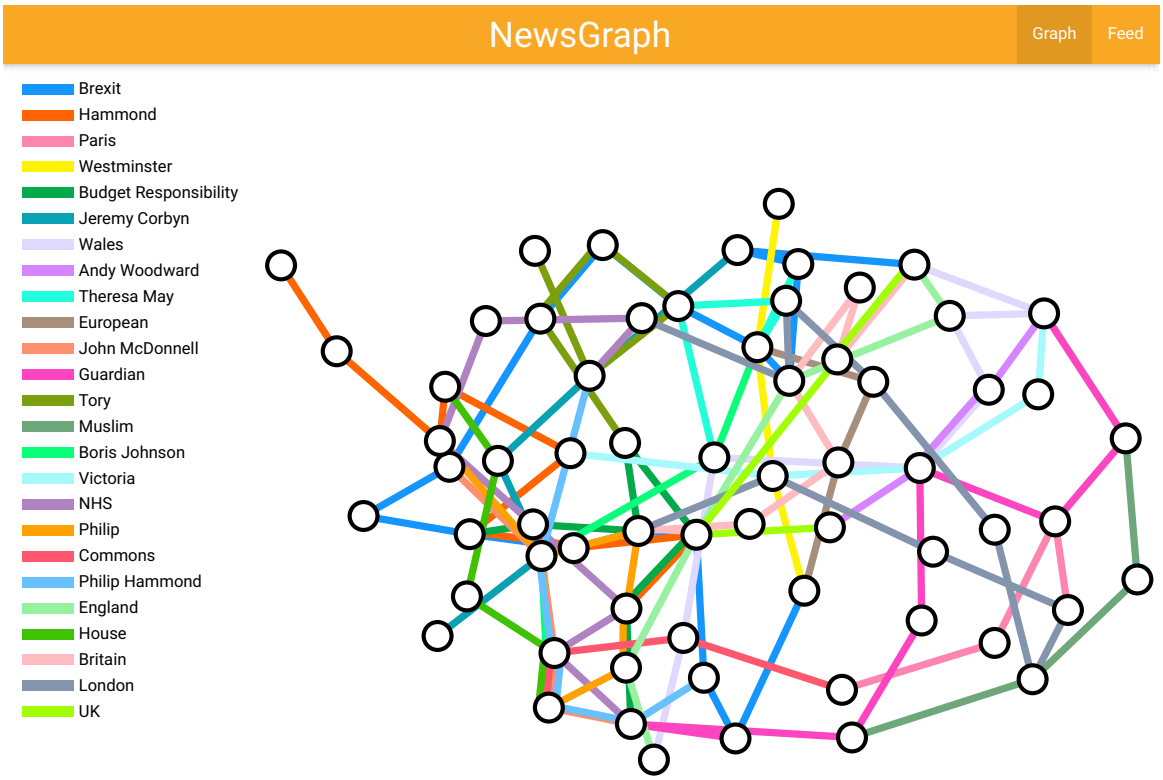


Figure 3.4: An unpruned and unusably dense metro map generated by the system

To do this, we first recall the original definition of document coverage (equation 1.1 [Shahaf et al., 2012*b*,*a*, 2013]), where d is a document, w is a keyword, and $\mathcal{W}$ is some normalised measure of term-frequency[10], such as tf-idf.

$$cover_d(w) : \mathcal{W} \rightarrow [0, 1] \tag{1.1}$$

In order to compare candidate metro lines, we build on this definition of coverage to extend it to paths (Equation 3.1).

$$Coverage(\mathcal{P}) = |\mathcal{P}| \sum_{d \in \mathcal{P}} \frac{cover_d(\mathcal{P})}{|\{\, p \in \Pi \mid d \in p,\, p \neq \mathcal{P}\}|} \tag{3.1}$$

Here, the extent to which a path $\mathcal{P}$ covers a document d within a corpus $\mathcal{D}$ is proportional to its tf-idf coverage of that document and inversely proportional to the number of other candidate paths which *also* cover that document. The coverage of the entire corpus by $\mathcal{P}$ is then given the sum of its coverage of the articles along it multiplied by the length of the path, with high coverage being desirable.

It is important to note that although with the final multiplication we show preference to longer metro lines, the objective of the system is not simply to maximise the number of articles which are included on the generated maps. This could easily be achieved by choosing the most nonspecific universal keywords as metro lines, but the resultant lines would be vague and unhelpful to a reader. It is in the nature of summarisation that not all information can be preserved, and in doing this kind of graph selection, the information lost is those articles which don't form links to any significant topics within the corpus.

3.4.2 Penalising Affinity

The principle of topic connectivity was the basis for choosing the metro map visualisation. Without it, we have simply generated a set of two-dimensional timelines with no contextual links. However, maps which are overly connected will quickly become unusable. In particular, maps where multiple lines runs adjacently through more than two nodes (see figure 3.5) are a sign of poor line choice. This is not simply hyper-connectivity, but correlation between two or more lines, often resulting from keywords which are semantically close.

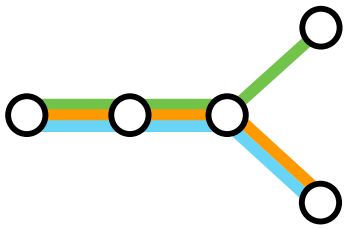


Figure 3.5: An example of three metro lines, all with high affinity

[10] In practice however, we found this normalisation to be unnecessary.

We call this function of two lines their *affinity*, and define the affinity of a single line as the sum of its affinities over the set of paths, excluding itself.

$$Affinity(\mathcal{P}) = \frac{1}{|\mathcal{P}|} \sum_{p \in \Pi} 2^{|\{\, d \,|\, d \in \mathcal{P} \cap d \in p, p \neq \mathcal{P}\}|} \tag{3.2}$$

The affinity of two lines is defined as the number of articles they have in common, raised as a power of two. The effect of the power is illustrated in Figure 3.6. On the right, line a (orange) which shares one article with line b (green) and another with line c (blue) does not indicate as strong a correlation as on the left, where a shares two articles with b and none with c. As with line coverage, we penalise shorter lines.

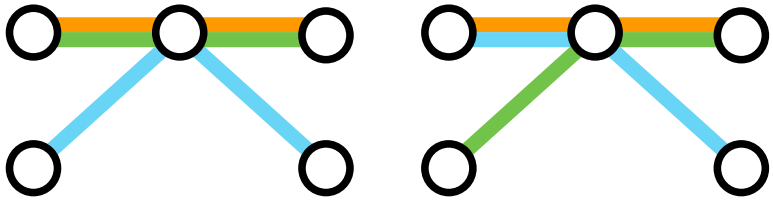


Figure 3.6: 'Run-on' affinity (left) is penalised more heavily than 'one-off' affinity (right)

Since metro lines should have both high line coverage and low affinity, a score is computed for every path by dividing the two composite measures (Equation 3.3).

$$LineScore(\mathcal{P}) = \frac{Coverage(\mathcal{P})}{Affinity(\mathcal{P})} \tag{3.3}$$

Sorting the list of candidates by descending score and filtering out those candidates which contain too few articles to be beneficial, the system can can then take the top n from the range $7 \leq n \leq 12$ to become the metro lines on the map. Any article which contains the name of a metro line in its keyword vector will be represented as a station on that line, and articles on more metro line than one will be represented as interchanges.

3.4.3 Coherence

The last metric proposed by Shahaf et al. is the one which presented the greatest conflict with the predefined scope of the system. For all but the most rapidly unfolding stories, forming a coherent chain of articles on one topic or event is not possible when only considering a single day's news.

Shahaf et al. had the advantage of assuming any topic well-known enough to be queried against their system would most likely be made up of a coherent chain of events, such as the Greek debt crisis or Brexit. In contrast, at a daily summarisation level, often no such chains exist; many articles exist in isolation or instead represent the *beginning* of a potential future storyline. Those articles which are part of a long-running chain of events are often still unpredictably published, and these chains of events can go weeks or months without

any resolution. Simply altering the parameters of the system to process a week's worth of news rather than a single day's would not solve the problem.

Given a large enough background corpus, it would be possible to link the day's summary back in time to older articles, extending metro lines further back to support the *zoom and filter* or *details-on-demand* processes. This is an area of great potential, as it could provide the story resolution the participants in the Associated Press and Context-Based Research Group [2008] study craved, allowing users to explore further back in time along individual metro lines while still centring itself around the most recent news. Unfortunately however, this is not something we could have achieved within the assigned timescale.

3.4.4 Serialisation

The intermediate stage of the pipeline between graph building and map drawing is purely an issue of implementation, with few design choices to be made. Starting with the Python representation of the metro map, articles are grouped by metro line to be serialised, along with their metadata and summaries, to JSON. This JSON is then copied into a preexisting templated JavaScript file which is responsible for representing and drawing the graph using D3.js. The contents of the JavaScript file are then embedded as a script in an HTML file, which contains the markup for the page.

The main advantage to this process is that it results in a single file, meaning the resultant visualisations are as portable, shareable and platform independent as the news they were derived from. A secondary advantage is that the JSON accepted by D3 is standard and could easily be passed a different JavaScript graphing library in the case of a more suitable alternative being written, with no change to the core Python software. The core software itself could also be hosted as a web app which serves the map HTML files, removing the need for any client-side processing or installation.

3.5 Map Drawing

The metro map layout problem, that is, determining whether an optimal planar embedding exists for a map and a set of hard constraints which include drawing straight octilinear lines, was proven to be NP-Hard by Nöllenburg and Wolff [2005]. The authors present a mixed-integer program (MIP) approach to metro map layouts, which both guarantees an octilinear layout and avoids falling into local maxima (unlike Stott's), but not in polynomial time.

At the time of writing, there are no public domain algorithms or open-source libraries for generating automatic metro map layouts, in any language. Those which have been proposed in academia [Stott, 2008, Nöllenburg and Wolff, 2005] are too complex to implement given the current timescale, and all require NP-Hard optimisation techniques. To overcome this, a heuristic approach derived from the work of [Stott, 2008, Stott et al., 2011, Purchase, 1997] will be detailed.

In this section of the pipeline, we are no longer concerned with the contents of articles or the entities they reference. The map drawing context marks a shift in the terminology from the previous stages of the pipeline to the domain of graph drawing, but many terms have a one-to-one correspondence. A list of definitions is given below, and in addition we recall Shahaf et al.'s definition of a metro map from Chapter 1, which ties the terms together.

Definition 1. *A metro map $\mathcal{M}$ is a pair (G, Π), where $G = (V, E)$ is a directed graph and Π is a set of paths, or metro lines in G. Each $e \in E$ must belong to at least one metro line.*

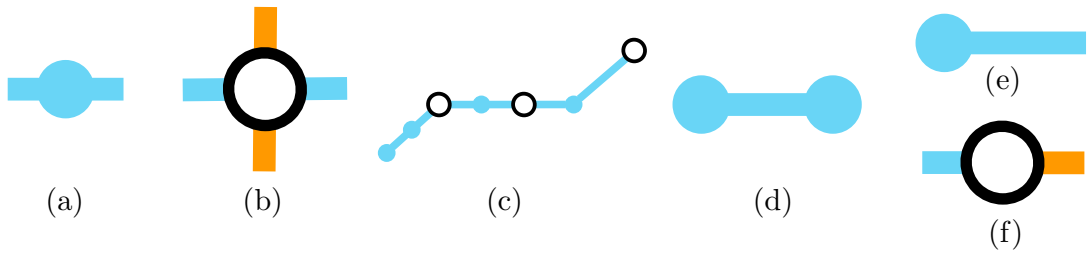


Figure 3.7: Left-to-Right: A station (a), an interchange (b), a metro line (c), a link (d), a terminus (e) and a non-terminus station (f).

Station A single article $v \in V$, represented by a node in the graph. If v has more than two incident edges, i.e. it represents article which spans more than one topic in the graph, it is called an *interchange*.

Metro Line An unbroken named path $p \in \Pi$ which is incident to two or more nodes, represented by a line of single colour. Metro lines are composed of *links*.

Link A direct edge $e \in E$ between two nodes, which does not pass through any other node. In contrast to Shahaf et al.'s definition, here, links must belong to *exactly one* metro line. The case of two lines running side-by-side through the same two nodes is represented by two distinct links. This difference is due only to an implementation detail of the graphing library chosen.

Terminus A node with only one incident edge. This is not the definition of a terminus in the sense of the underlying metaphor, since by our definition, a station which is the terminus for two lines (see Figure 3.7.f) is no longer categorised as a terminus at all. Instead, the definition requires that the position of a terminus only determine the position of a single link in the graph.

We now turn our attention to transforming the metro map structure into a planar embedding of stations and the metro lines which connect them. As we have previously observed, not all graph layouts were created equal, and the aesthetic quality of a graph-based visualisations has a notable influence on their usability and ease of interpretation [Purchase et al., 1997, Purchase, 1997, Stott, 2008, Stott et al., 2011].

In Stott's words: "[The] visualisation process is deeply rooted in the human interpretation of the graph as a structure and as such, the quality of the aesthetics of a particular drawing of a graph are very important." [Stott, 2008, p.24]

We therefore devote a significant amount of thought to optimising the visual quality of the generated metro maps by revisiting Stott's aesthetic criteria for layouts, discussed in Chapter 1. We justify which of these apply to our visualisations, and for those which are, we detail the methodology used to implement them.

3.5.1 Stott's Criteria Metro Map layouts

The set of rules and criteria defined by Stott [2008] form a complex multicriteria optimisation function. Starting with an initial layout, nodes are embedded on a square grid of finite size, and iteratively moved to maximise the node movement criteria, as long as movement does not violate one of the node movement rules.

It should be noted that the function also applies to the positioning of labels and the maximum of a set of label movement criteria alongside the node movement criteria, but as our stations are unlabelled, these will not be described.

As previously discussed, there are six node movement criteria, which – using the metro map-specific terms defined above – are as follows:

1. **Angular Resolution**: Maximise the angle of incident links on the same metro line at each station. As Purchase [1997] found satisfying this criteria had no statistical effect on task performance, this will not be a priority[11].
2. **Edge Length**: Keep all links approximately the same length. This is a criterion which can be implemented via a soft constraint using the graphing library, but it will not be considered beyond this.
3. **Balanced Edge Length**: Keep links incident to a given station approximately the same length. While it would be possible to implement this as an additional constraint, there is no evidence that it would significantly improve usability, so this will not be considered.

[11] However, as angular resolution comes as a natural consequence of maximising line straightness, it is not disregarded completely during implementation.

4. **Edge Crossings**: Minimise the number of link crossings. Purchase found this to be the most significant factor in determining task performance, therefore it will be considered.

5. **Line Straightness**: Maximise the collinearity of links on the same metro line. Purchase found this criterion to be somewhat beneficial to task performance, so this is both a realistic and justified constraint which will be considered.

6. **Octilinearity**: Draw links at multiples of 45°. Although Purchase found orthogonal graphs to be insignificant factors on task performance, there is a strong argument that the familiarity of the metro map metaphor hinges on certain similarities being preserved. Octilinearity is a noticeable design feature in the metro maps of the world, and therefore should be attempted in our visualisations.

Based on the justifications given above, we consider minimising edge crossings to be the highest weighted criterion, followed jointly by maximising octilinearity, then preserving line straightness.

3.5.2 Initial Force-Directed Positioning

A major difference between the maps generated using this system and maps representing real transit networks is the lack of "real" starting positions we have for stations. While the freedom to move points around without having to preserve their relative positioning removes one layout constraint, finding a set of starting positions for the map layout which define a planar embedding with minimal line crossings is non-trivial.

As Stott notes, "This [difficulty] is due to our method being based on optimising an existing layout: if the initial embedding is not adequate, then our method may struggle to produce an acceptable optimisation."[Stott, 2008, p.210].

One solution Stott suggests is to generate positions using a force-directed algorithm; a class of algorithms which simulate the forces of real physical motion to position nodes in a graph, with minimal edge crossings and approximately uniform edge lengths [Kobourov, 2012]. The drawback to using a force-directed approach is that initial positioning and therefore the final map layout becomes nondeterministic.

D3 includes an optimised implementation of Dwyer's [2009] force-directed layout algorithm, and provides several parameters which can be used to alter the forces applied to some or all of the nodes in the graph. The two forces we will adjust to optimise the force-direction for metro maps are as follows:

- `force.charge([charge])`[12]

  Charge specifies the force of attraction between nodes, with negative values causing nodes to repel each other. In order to minimise crossings, this should be a large negative value, particularly for nodes which of higher degrees, which will be surrounded by more edges. We therefore set charge to -300 for termini, and -500 for all other stations.

[12] `https://github.com/d3/d3-3.x-api-reference/blob/master/Force-Layout.md#charge`

- `force.alpha([alpha])`[13]

  Alpha is the graph's cooling parameter, decaying as the simulation converges on a stable layout. As the temperature falls, node movement slows down, and eventually alpha will drop below a threshold which pauses the simulation, preventing further movement.

Unfortunately, the implementation of `d3.layout.force` lacks any further mechanism for detecting or reducing line crossings due to the computational expense of the best known algorithms (this subproblem is also NP-Hard). Even if line crossings could be detected, it is likely that the topology of certain graphs generated would make it impossible to find a planar embedding with no crossings.

The parameter `force.alpha` is re-evaluated every time the simulation is advanced by a single step, which happens approximately 60 times per second. Since our aim is to generate an initial layout with minimal line crossings which will result in the best possible map, the simulation can constantly vary `force.alpha` based on the quality of the map as its stations are repositioned.

This means that if the stations settle into a map layout with desirably low octilinearity and line straightness, the energy of the graph will be reduced such that it is impossible for the embedding to change. Likewise, if the stations start repelling each other in a way that worsens line straightness and octilinearity, the system will have its kinetic energy increased and will be more likely to move out of the undesirable state.

```
98  force.nodes(graph.nodes)
99    .links(graph.links)
100   .on("tick", function() {
101       // Update the energy map's energy according to the aesthetic
102       // criteria. Poor aesthetics => More energy to reposition.
103       energy = Math.log(octilinearity())/10 +
104                Math.log(lineStraightness())/10;
105       force.alpha(energy);
106       tick();
107   })
108   .start();
```

Listing 1: Recalculating `force.alpha` from Stott's [2008] criteria: `newsgraph.js`

The measure of quality we use to update the kinetic energy of the stations is a combination of the two remaining chosen aesthetic criteria; octilinearity and line straightness, as shown in Listing 1. Strict definitions for these criteria and the means by which they are calculated are provided in the following two sections. Separate from the map octilinearity criterion but using the same principle is Section 3.5.3.1, which describes the repositioning logic for the outer sections of metro maps.

[13] `https://github.com/d3/d3-3.x-api-reference/blob/master/Force-Layout.md#alpha`

3.5.2.1 Octilinearity

The octilinearity criterion (Equation 3.4 [Stott, 2008]) penalises lines which are not drawn at multiples of 45° angles; the further the gradient of a link in the graph is from a multiple of 45°, the higher its octilinearity. Lower values are desirable, with zero indicating a graph is entirely octilinear.

$$\text{octilinearity}(G) = \sum_{u,v \in E} \left| sin\ 4 \left(tan^{-1} \frac{|u.y - v.y|}{|u.x - v.x|} \right) \right| \tag{3.4}$$

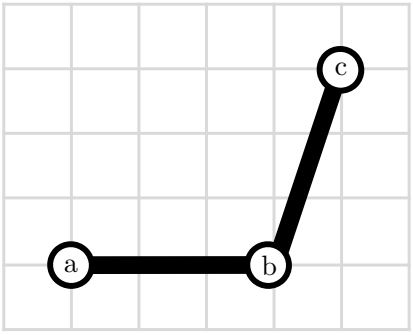


Figure 3.8: Octilinearity Calculation

The octilinearity of the graph in Figure 3.8 (left) can be calculated as follows, taking positions relative to $(0, 0)$ in the bottom left corner. Algorithm 3 describes this calculation.

$$\text{octilinearity}(a, b) = |sin\ 4\,(tan^{-1} \frac{|1 - 1|}{|4 - 1|})| = 0$$

$$\text{octilinearity}(b, c) = |sin\ 4\,(tan^{-1} \frac{|4 - 1|}{|5 - 4|})| = 0.96$$

$$\text{octilinearity}(a, b, c) = 0 + 0.96 = 0.96$$

There is a caveat to this formula which goes unmentioned by Stott, but which is necessary to emphasise as a detail of implementation. In the case where $u.x - v.x = 0$, we take octilinearity to be zero, as this value indicates a straight vertical line. Otherwise, attempting to calculate the inverse tangent of a fraction whose denominator is zero (though this is mathematically valid, as the principle value of $\tan^{-1} \infty$ is $\frac{\pi}{2}$) would result in undefined behaviour.

Algorithm 3: Calculating map octilinearity

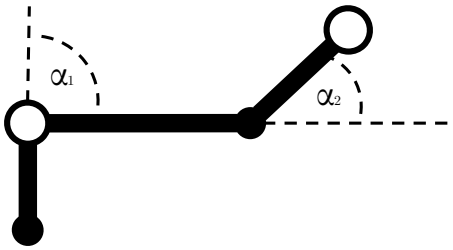
Data: $\mathcal{M}$: a metro map $(G = (V, E), \Pi)$
Result: $oct \in \mathbb{Q}, 0 \leq oct \leq \frac{24}{25}|E|$: the octilinearity of $\mathcal{M}$

```
1  oct ← 0 ;
2  foreach e ∈ E do
3  |   δ_x ← |e.source.x − e.target.x| ;
4  |   δ_y ← |e.source.y − e.target.y| ;
5  |   if δ_x = 0 ∪ δ_y = 0 then
6  |   |   slope ← 0          /* This avoids a potential division by zero.  */
7  |   else
8  |   |   slope ← δ_y / δ_x
9  |   end
10 |   θ ← |sin(4 × tan^-1(slope))| ;
11 |   oct ← oct + θ ;
12 end
```

3.5.2.2 Line Straightness

The line straightness (Equation 3.5 [Stott, 2008]) criterion is defined as the sum of every 'turn' a metro line makes between its first and last stations, where a turn is measured as the angle between the new link direction and the continuation of the previous direction. The global line straightness is simply the sum of the straightness of every metro line in the map, and as with octilinearity, lower values are preferred.

$$\text{lineStraigtness}(G) = \sum_{p \in \Pi} \sum_{u,v \in p} \theta(u, v) \tag{3.5}$$



The line straightness of the graph in Figure 3.9 (left) is therefore $\alpha_1 + \alpha_2$, where the values of α_1 and α_2 can be found geometrically from the direction of the links (Algorithm 4).

Figure 3.9: Line Straightness Calculation

Algorithm 4: Calculating map line straightness

Data: $\mathcal{M}$: a metro map $(G = (V, E), \Pi)$
Result: $ls \in \mathbb{Q}, ls > 0$: the line straightness of $\mathcal{M}$

$ls \leftarrow 0$;
foreach $p \in \Pi$ **do**
  `/* Greedily iterate over all 3-tuples of adjacent stations on` p `*/`
  foreach $(a, b, c) \in p$ **do**
    $v_1 \leftarrow \frac{\overrightarrow{ba}}{|\overrightarrow{ba}|}$; $v_2 \leftarrow \frac{\overrightarrow{bc}}{|\overrightarrow{bc}|}$;`/* Normalise the two vectors to unit length */`
    $\theta \leftarrow cos^{-1}(v_1.x \times v_2.x + v_1.y \times v_2.y)$;`/* Find the angle between links */`
    $ls \leftarrow ls + \theta$;
  end
end

Empirically, the best results came from weighting octilinearity and line straightness equally when using them to set `force.alpha`, but this could be refined through further investigation.

The actual value assigned to `force.alpha` is the sum of the natural log of both values, to ensure that small improvements to already low values for either criterion are rewarded. This sum is also divided by a constant, due to the extreme responsiveness of the force-directed algorithm to large values of `force.alpha`.

3.5.3 Heuristic Refinement

Once all stations have been assigned their starting positions, they are incrementally moved to adjacent intersections, according to a set of conditions for improving the quality of the map. The first step is to discretise the search space for station positions, by *snapping* stations to intersections on a large grid. We found the optimal size for this grid to be in the order of $\frac{n}{4} \times \frac{n}{4}$ for a map containing n stations.

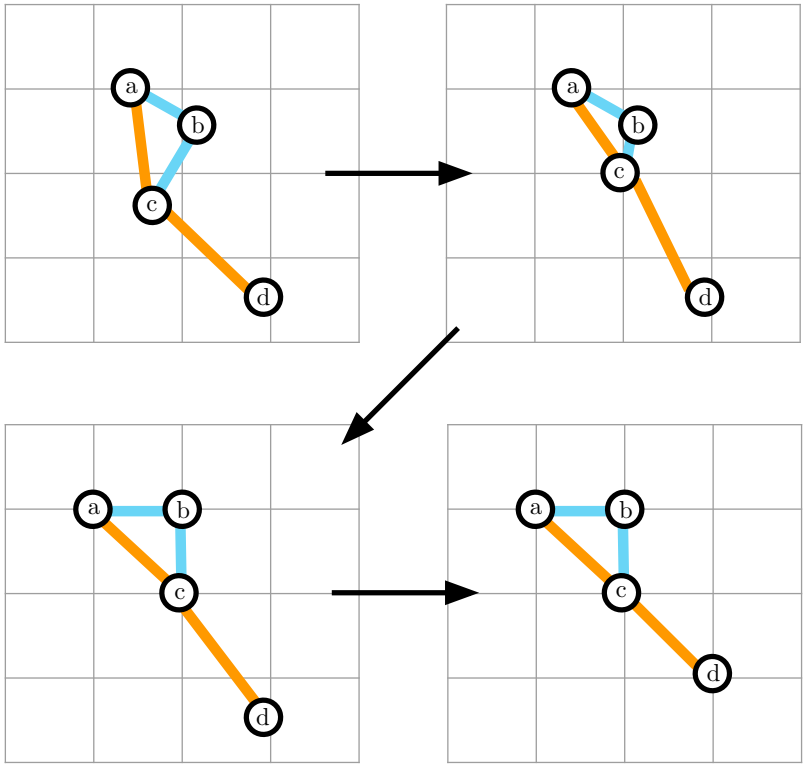


Figure 3.10: An example set of initial moves demonstrating the *snap-to-grid* process

Figure 3.10 shows the *snap-to-grid* process applied to a small example. Stations are 'snapped' in order of descending weight (number of incident edges), as typically the more edges a node is connected to, the more difficult it is to place. In the example, this means station c (with weight three) is moved first. There is contention for the second move, as both stations a and b have weight two, so we make an arbitrary decision as to which we move first, and the two moves are shown as one step. The final station to move is station d, after which we can mark all stations as successfully placed.

Stations whose nearest intersections are already occupied are left unplaced. This implementation contrasts slightly with Stott's approach, where stations are snapped *in order* of Euclidian distance to their nearest intersection, and snapped to the next-nearest if the closest intersection is already occupied. This is because Stott's implementation performs poorly for maps with several highly connected nodes, which is a common attribute of metro maps of news, due to the publishing of review articles which often span many of the lines on the map.

To assign the final station positions, our approach now diverges from Stott's into a heuristic method which has been refined for metro maps comprising between 10 and 50 stations. It is vital that lines with high affinity have been removed before this stage, or the following process will result in a map where nodes with high weights are occluded.

Although by definition the *snap-to-grid* gives a map near-perfect octilinearity, it can significantly worsen line-straightness and several of the other aesthetic criteria, as shown in Figure 3.11. It would be preferable to have the circled station repositioned as the transformation shows, even though this sacrifices the uniform edge length of the map. It is therefore necessary to make several passes over each metro line, moving stations to the midpoint of the line between the previous and next neighbouring stations if that line is octilinear.

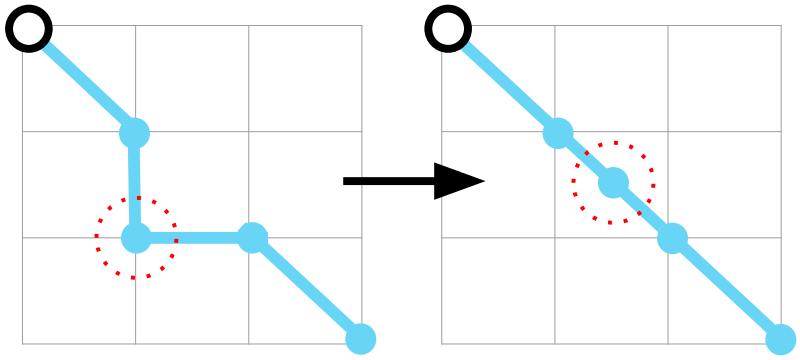


Figure 3.11: An octilinear move which compromises the edge-length criterion

In addition, a final pass checks for stations which are still unplaced at this point and performs the same midpoint movement regardless of whether the line between the next and previous points is octilinear. This is an attempt to ensure the unplaced point obscures and crosses as few other links as possible, in what is most likely a densely packed area of the map. An example of this process is shown in Figure 3.12.

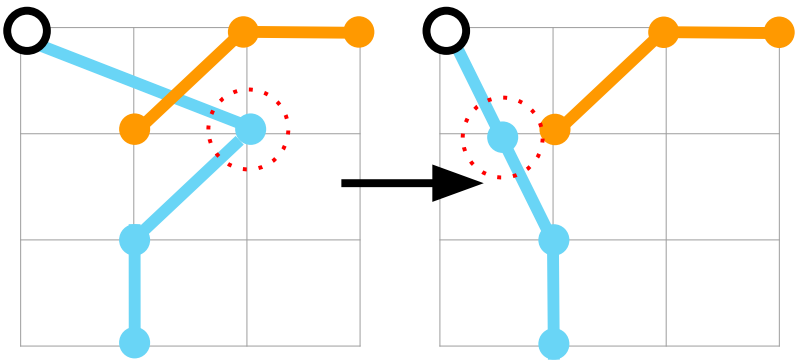


Figure 3.12: A non-octilinear move of an unplaced station

3.5.3.1 Octilinearising Terminus Branches

The three-step process for station movement described above focuses on repositioning the more connected portions of the metro maps generated by the system, which are typically found in the centres of the maps. This process does not help shape *terminus branches*, that is, portions of metro lines which lead to or from its terminating stations and which have no intersections with other metro lines (see the blue metro line in Figure 3.13).

The nature of D3's force-directed algorithm and our lower `force.charge` rating for terminus stations causes straight lines to curve under the force, resulting in terminus branches being snapped to grid intersections which approximate the curve, as shown in figure 3.13, rather than as straight lines.

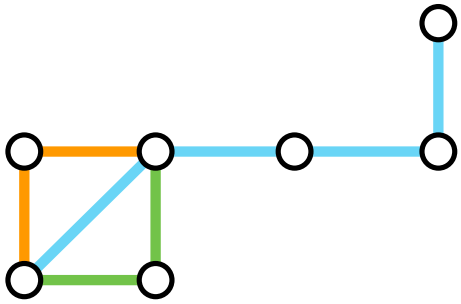


Figure 3.13: A terminus branch containing an unnecessary bend

Terminus branches are often easy to reposition, because there is less contention for space outside the centre portion of the map. They are also not hard to identify; stations which fall between termini and any intersections can be accumulated as part of a branch, and this accumulation continues until an intersection with a different metro line is reached.

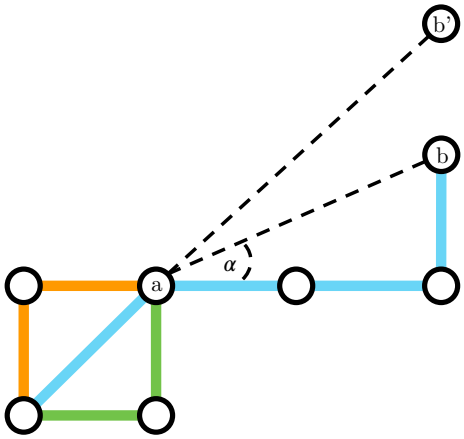


Figure 3.14: The candidate move for the branch in Figure 3.13

The repositioning process is illustrated in Figure 3.14. The gradient α of line $\overrightarrow{ab}$ is 22.5°, which, rounded to the nearest octilinear angle is 45°. The new location for the terminus station b is a line of length $|\overrightarrow{ab}|$, at an angle of 45°, represented in the figure by the dotted line $\overrightarrow{ab'}$.

The JavaScript implementation of this function shown in Listing 2 works bidirectionally, on branches which both lead *to* a terminus or originate *from* a terminus, according to the chronology of the line. This function performs the exact process shown in Figure 3.14, by finding the nearest octilinear angle to the line between the stations `begin` and `end`, and returning an object containing the new co-ordinates of the stations. From the co-ordinates of the two branch endpoints, the positions of all the stations between can be interpolated in a separate function.

```
328 function octilineariseLine(metro_line, begin, end, dir) {
329   let l = metro_lines[metro_line];
330   let line = {x: end.px-begin.px, y: end.py-begin.py};
331
332   let alpha = Math.atan(line.y/line.x);
333   let nearest = Math.ceil(alpha/(Math.PI/4)) * (Math.PI/4);
334
335   if (dir == 1) { // line coming inwards; move 'begin' station
336     if (Math.round(Math.abs(nearest),2)==2 || nan(Math.tan(nearest))) {
337       begin.py += linelength(begin, end);
338     } else {
339       begin.py += line.x * (Math.tan(nearest)-Math.tan(alpha));
340     }
341   } else { // line going outwards; move 'end' station
342     if (Math.round(Math.abs(nearest),2)==2 || nan(Math.tan(nearest))) {
343       end.py += linelength(begin, end);
344     } else {
345       end.py += line.x * (Math.tan(nearest)-Math.tan(alpha));
346     }
347   }
348   return {b:begin, e:end};
349 }
```

Listing 2: The `octilineariseLine` function for terminus branches: `newsgraph.js`

Position b' is therefore chosen as a candidate move for station b in Figure 3.14. The stations on the branch between the two points are spaced evenly along the line $\overrightarrow{ab'}$, the result of which is shown in Figure 3.15. In the case of an orphaned metro line with no intersections, the result of this process is that the entire metro line will be drawn at an octilinear angle, with its stations evenly spaced between its two termini.

As a final step in the *map drawing* stage, octilinearity and line-straightness are recomputed for the repositioned map, and the number of points left unplaced (i.e. drawn directly on top of another point) are counted. If any of these the numbers are high, the process can be restarted and the randomness of the initial force-direction will result in a different initial embedding. The exact value of 'high' is a dependent on the number of articles on the map and has not been sufficiently investigated. Perfect octilinearity and line-straightness are typically only achievable for small or very sparse metro maps, as the more stations a map contains, the more likely it is that an article will be left unplaced.

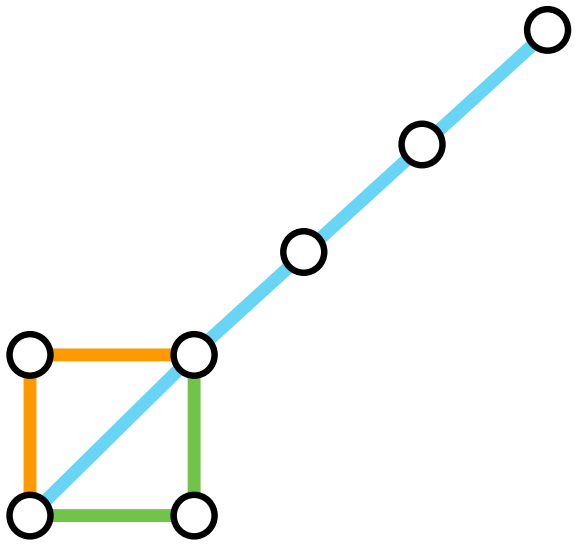


Figure 3.15: The result of octilinearising the terminus branch in Figure 3.13

3.6 Summary

We have given a detailed description of the system's pipeline, covering transformation of news data from XML elements to stations on an interactive metro map. We consider the following processes to be novel or significant successes of development:

1. Entity disambiguation through substring matching, denominal adjective stemming, and Google Knowledge Graph result ratios (Section 3.3).
2. Formalisation of the metrics *Path coverage* and *Line affinity* (Sections 3.4.1 and 3.4.2).
3. The first application of Stott's [2008] aesthetic criteria to metro maps representing news corpora (Section 3.5).
4. A heuristic function to improve the layout of terminus branches within small ($n < 50$) metro maps, by octilinear repositioning of termini (Section 3.5).

Several unexpected difficulties arose during the implementation of the system, in particular the tradeoffs involved with the certainty of entity disambiguation, quantifying the logic for determining what makes a "useful" metro line, and deciding which metro lines to select between two or more with shared high affinity.

The most significant challenge of implementation was undoubtedly the process of drawing and refining usable metro maps. The conflict between edge crossings, line-straightness and octilinearity was always at play, and the final algorithm still falls into both local and global minima as it is unable to detect the edge crossings. This section of the pipeline would benefit greatly from further work, as it is a significant area of active research in its own right.

As the expected datasets for the map are typically small (with fewer than 50 stations), a non-polynomial time algorithm for refining station positions would be feasible in terms of running time, and would doubtless improve the quality of the maps generated.

The results of the process documented in this chapter are illustrated in Figure 3.16. This example is available to interact with as `example-metro-map.html` in the `src` directory of this project.

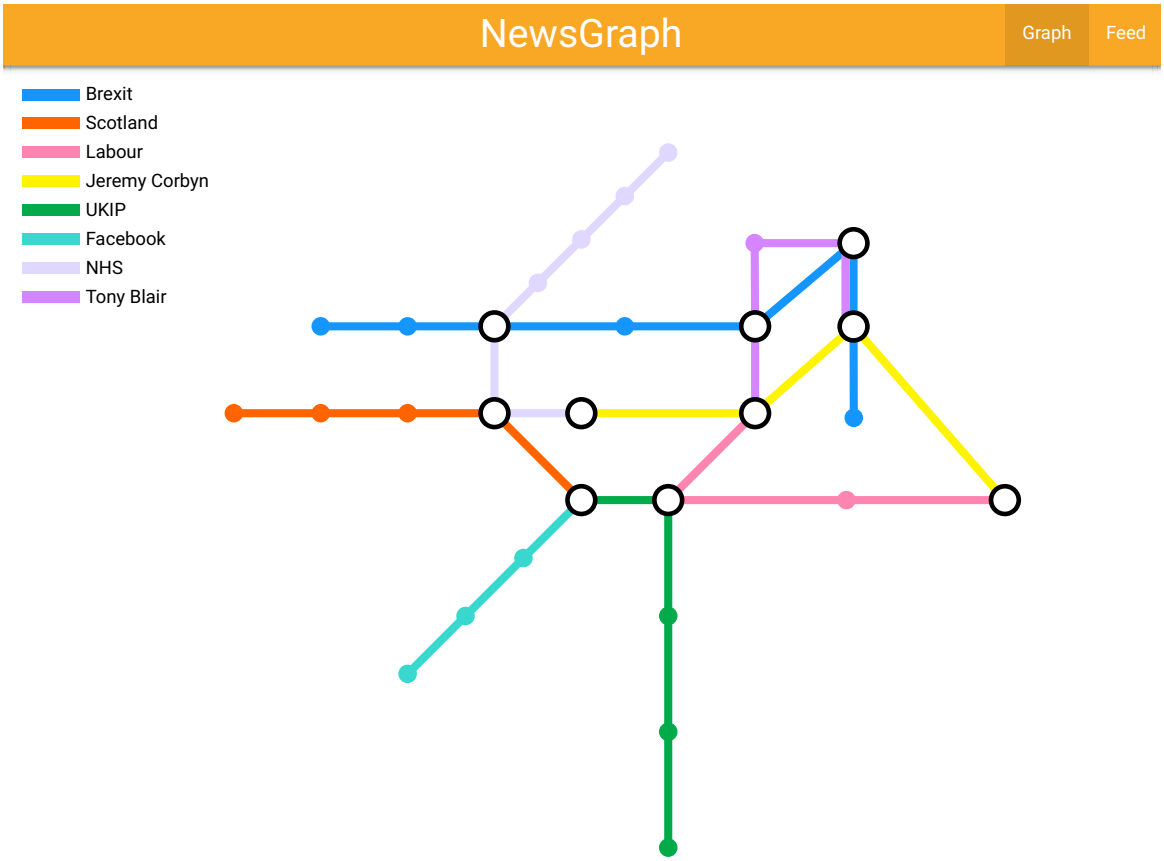


Figure 3.16: A metro map generated using the system

Further results obtained by using the system to generate metro maps for various news corpora are presented in the next chapter.

Chapter 4
Results and Discussion

This chapter presents the results of applying our method to a selection of RSS feeds spanning various news categories from well known news;

- **Politics** from *BBC News*, where we attempt to determine whether the map provides good coverage of the day's main stories (Section 4.1).
- **Business** from *The Telegraph*, where we discuss best and worst case map layouts including common pitfalls (Section 4.2);
- **Sport** from *The Guardian*, where we provide a warning on the risks of misinterpretation and false inferences (Section 4.3).

4.1 UK Politics (BBC News)

The main topic threads for March 16th 2017 in British politics were Scottish First Minister Nicola Sturgeon's call for a second independence referendum, the government's U-turn on National Insurance increases for the self-employed in the Spring Budget, and The Queen granting Article 50 royal assent, therefore enabling Brexit negotiations to begin.

Figure 4.1: The front page BBC Politics article; drawn as (a) in Figure 4.2.

To determine whether the map captured day's important political issues, we turn back to the native representation of the news. The BBC selected a piece on the proposed Scottish referendum as the front page article for their Politics page (Figure 4.1); a position which is generally indicative of importance.

In contrast, RSS feeds have no notion of relative content importance, so the system has no way of knowing this article should be included. Nevertheless however, when the article is parsed, its keyword vector contains {`'Scotland': 1.09861`} as the top entity. This term-frequency was strengthened by the transformation of every instance of the denominal adjective *Scottish* into the noun form, *Scotland*. The concentration of articles which focus heavily on Scotland (its coverage score of 182.644 was the third highest of the corpus) means it is selected as a metro line, resulting in the article being added to the map as we had hoped (see Figure 4.2).

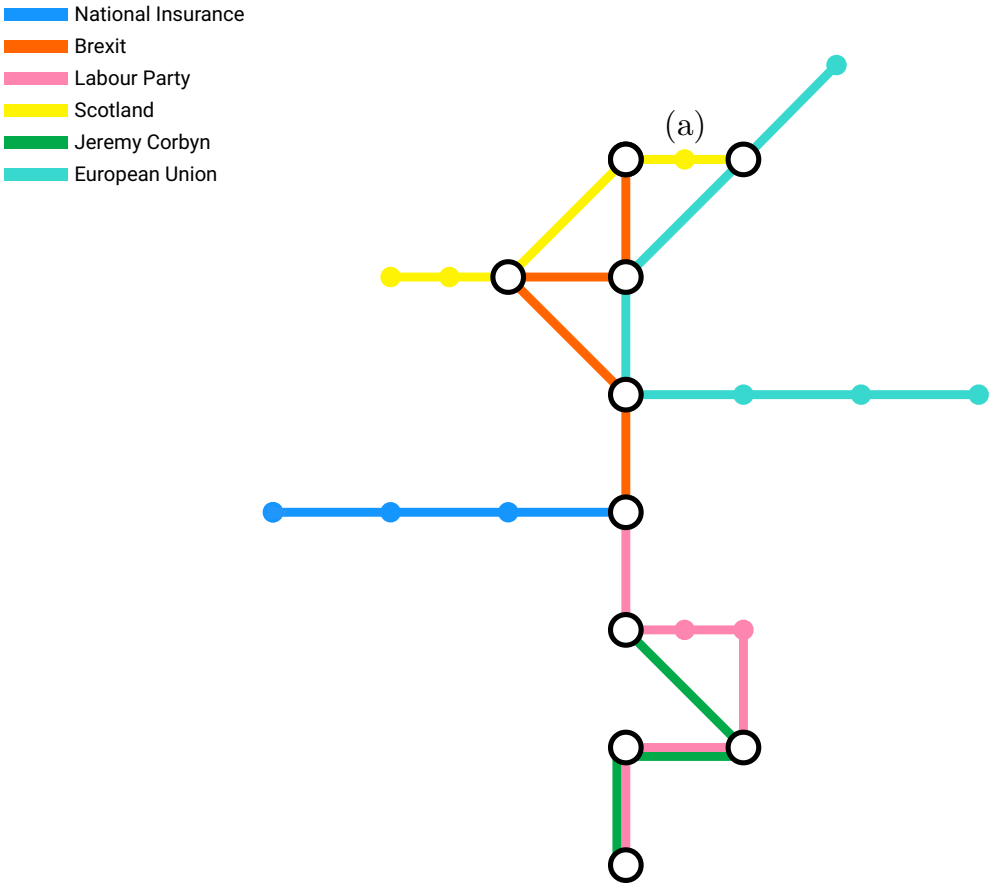


Figure 4.2: A metro map covering the BBC UK Politics RSS feed (March 16th 2017)

As well as a series of articles following the progression of the proposed referendum, the system also selects the other two main threads as metro lines. *National insurance* is selected despite its relatively short length due to its low affinity with the other lines. The system interprets this low affinity as the line covering a set of articles which no other line has, which boosts its likelihood of selection.

Despite being entirely subsumed by the *Labour Party* line, *Jeremy Corbyn* is still selected for the map because of the strength of its keyword score within the articles which contained it. Intuitively, we would hope that users realise lines which are entirely subsumed by other lines are important in their own right, while still being closely related to their subsuming entity.

4.2 Business (The Telegraph)

The system also performs well for international business news, where entities are typically the names of companies and organisations as opposed to people. NLTK Struggles with place names which have containment issues, for example, 'City of London' is split into *City* and *London*, and 'Bank of England' into *Bank* and *England*. *London* and *England* are then penalised by tf-idf for being common, but *Bank* and *City* are both selected as metro lines. In spite of this contrived error, the map is still largely usable and exhibits good coverage.

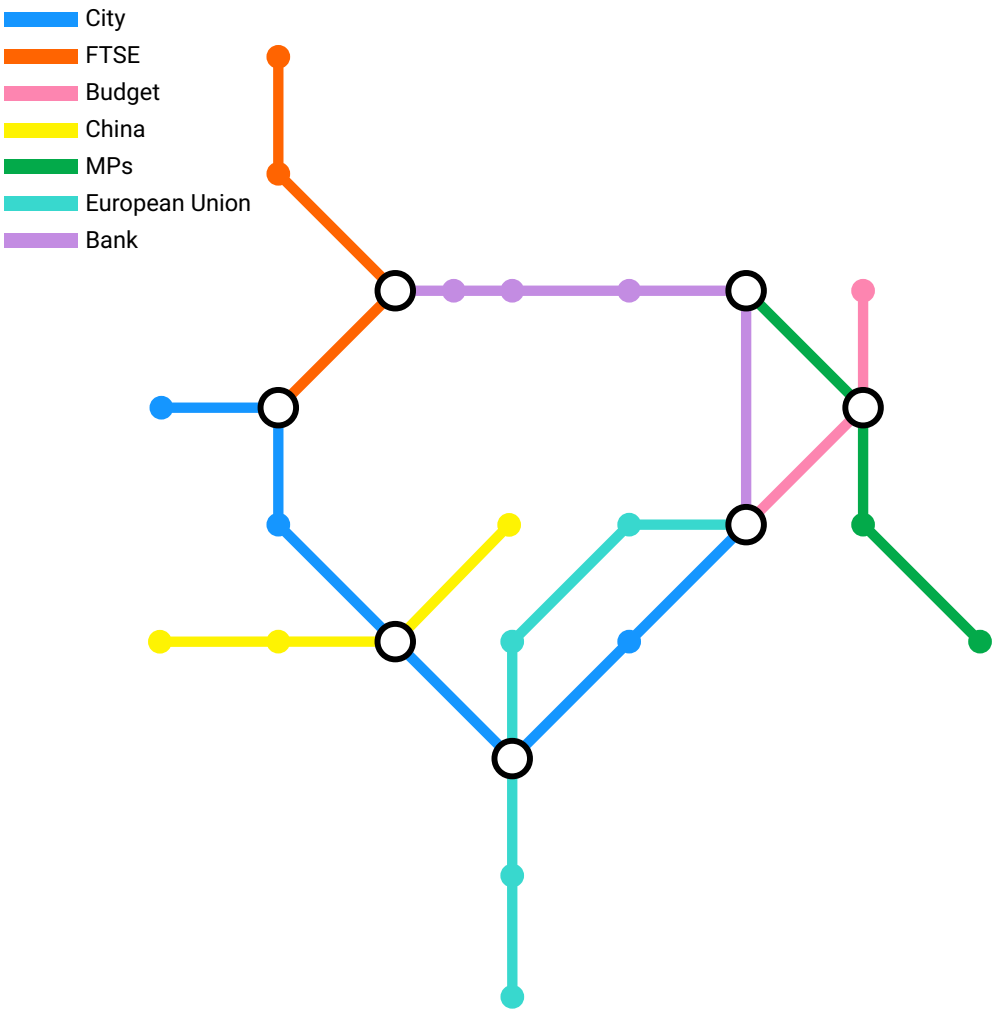


Figure 4.3: A metro map covering The Telegraph's Business RSS feed (March 17th 2017)

From an aesthetic perspective, a common pitfall of our approach is visible on the *Bank* line (Figure 4.3). During repeated passes along each octilinear line, stations are moved to the midpoint of their immediately adjacent neighbours, which results in noticeably variable edge lengths. A fix for this would be to iterate between pairs of stations which represent the endpoints of a run of stations all with weight two and distribute the accumulated stations at uniform intervals between the two points (Figure 4.4).

Figure 4.4: The redistribution of unbalanced edges along a line

A second pitfall is the inability of the algorithm to smooth the *European Union* line in Figure 4.5 between two points which are not octilinear to each other. The shape of the *China* line here is also frequently observed, because the initial force-directed algorithm causes lines to form circular arrangements, which are then snapped to co-ordinates which form squares.

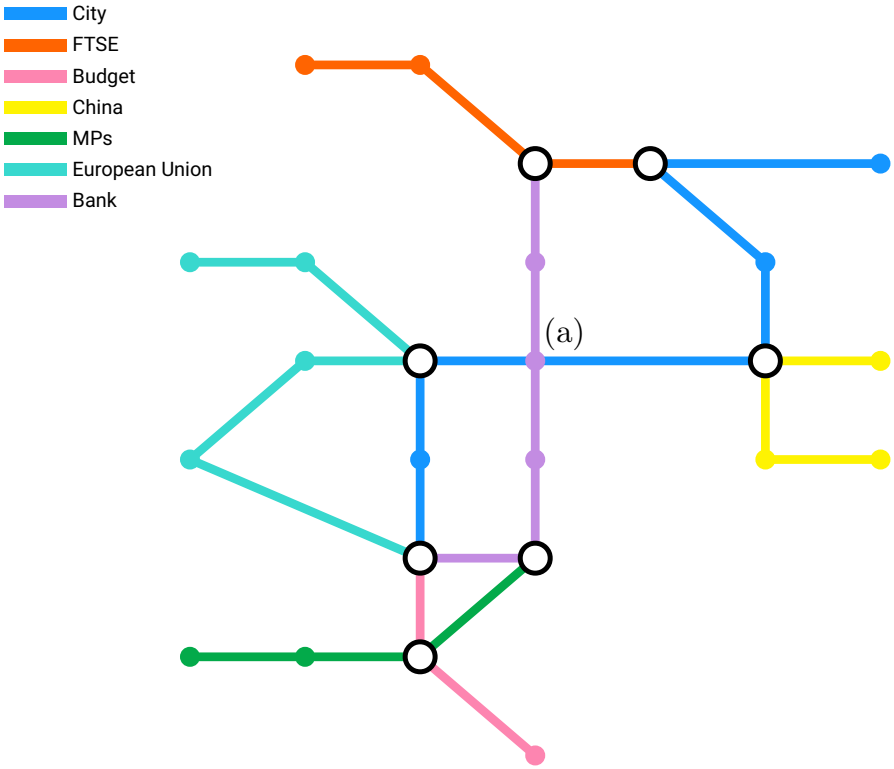


Figure 4.5: A poor embedding of the map in Figure 4.3

4.3 Sport (The Guardian)

Sports journalism was popular among the participants of the Associated Press [2008] study, in part because of the clean cut categorisation of teams within sports, within countries where applicable. Story resolution could be always achieved on a match-by-match basis, or over one or more seasons of predefined length. Current events journalism, in contrast, includes stories which are complex and sprawling in nature, often continuing over a period of days, weeks or even years.

Our system was designed for use with current events articles rather than sports content, but the results obtained through doing so bring to light some interesting issues which were not evident in previous examples, and therefore warrant discussion.

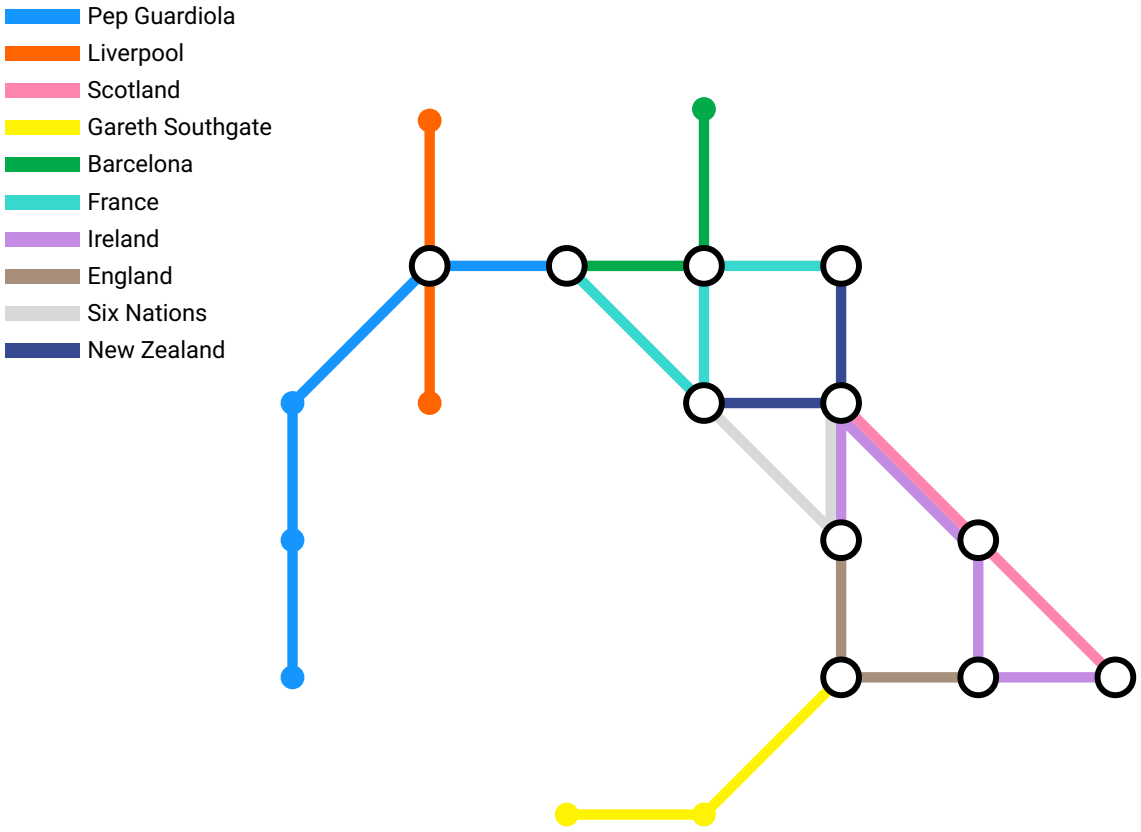


Figure 4.6: A metro map covering the Guardian Sport RSS feed (March 16th 2017)

If someone who knew very little about rugby were to look at the metro map on sports and see *Six Nations*, they would most likely (and correctly) assume that it is the name of a tournament with six participating countries. Using only the connections drawn on the map however, they may also assume that all five countries whose metro lines intersect with Six Nations are participants. This is incorrect; New Zealand, which intersects twice, is not.

This is a good example of a metro map which exhibits desirable aesthetic properties and

good coverage, but which is ambiguous and therefore uninformative at a thematic level. The names of the majority of the metro lines are countries, which create widespread entity ambiguity problems; does *England* imply the football team or the rugby team? The answer to this question is unfortunately both, as the system is unable to disambiguate. The risk of misinterpretation by a user leading to confusion and contributing further to information overload is of course ever present, but particularly noticeable in this example.

The quality of the metro lines on this map confirmed our suspicious that the system is not practical for use with corpora which contain a large number of ambiguous entities. That is not to say, however, that metro maps are inappropriate for all news feeds related to sport. Generating a metro map based on The Guardian's Football RSS feed (Figure 4.7) does not lead to the same team name ambiguities, and therefore results in an unequivocally distinct set of metro lines.

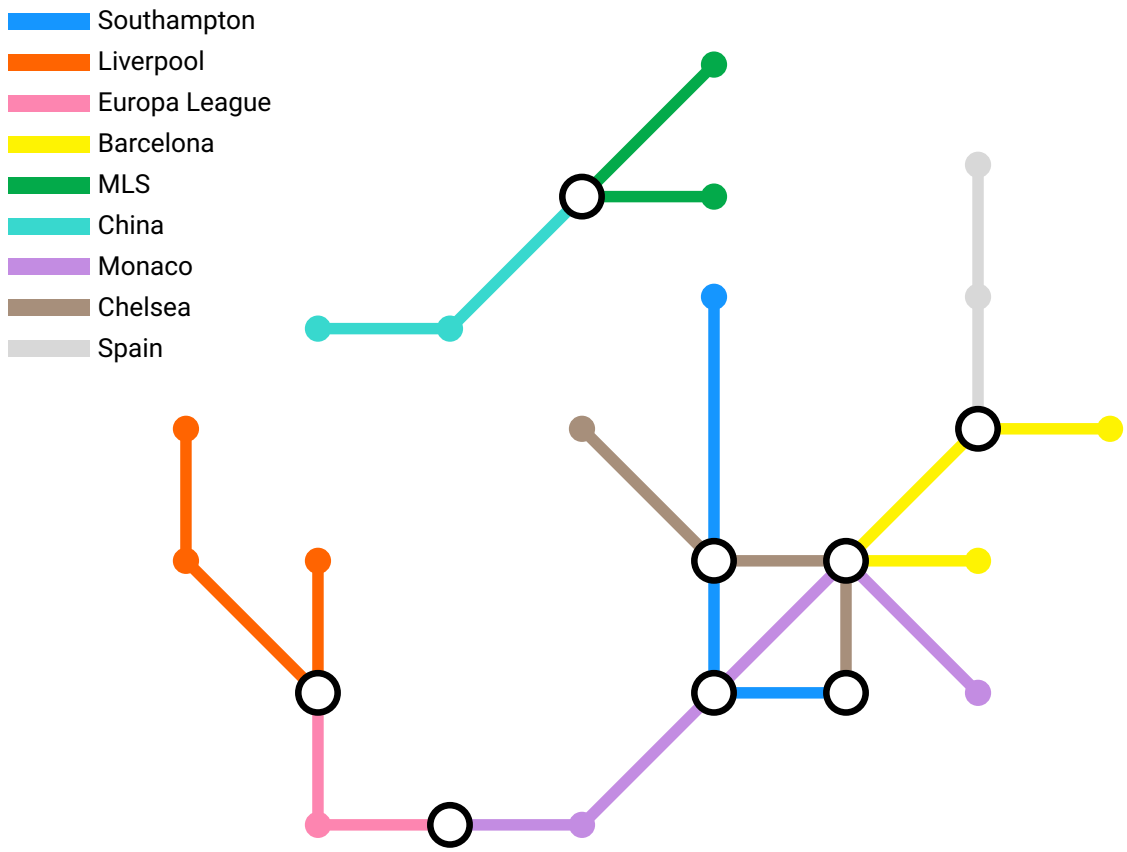


Figure 4.7: A metro map covering the Guardian Football RSS feed (March 17th 2017)

News coverage for individual sports and leagues, it transpires, can be well represented on metro maps. This is mainly due to keyword extraction being less complex in this domain; the names of most teams and sports personalities are easily identifiable with a knowledge base. To only select *teams* as metro lines, we could simply filter out other entities at the Knowledge Graph stage, using the `Thing > Organization > SportsOrganization >`

`SportsTeam` type from Schema.org. The same principle could be applied to countries or athletes, in order for users to adjust the schema represented by the metro lines at a *zoom-and-filter* level [Shneiderman, 1996].

Generalising this relationship between well defined schemas and a well defined set of metro lines to domains other than sports would suggest that in order to improve line selection for any map, it may be beneficial to restrict line choice to those which fit certain predefined types, such as `Thing > Person`, `Thing > Organization`, and `Thing > Place`. This kind of schematisation immediately leads to the question of which entities we should consider useful. The answer to this question may be domain-specific, user-specific, or most likely a combination of the two.

4.4 Summary

This chapter presented three different perspectives from which the results of the system can be analysed; by content, by layout, and by context.

Firstly, we verified that our maps provide sufficient coverage of topics which were deemed significant by the original news source, and discussed a concrete example of two topics being selected as metro lines despite their high affinity with each other.

Following on from content, we took a step back and evaluated the maps purely in terms of the success of their embeddings. Best and worst case layout examples were provided examples for a particular current events map, where we identified common weaknesses of the map layout algorithm, in particular the inability of the layout algorithm to smooth non-octilinear edges once stations have been marked as placed.

Finally, we used the system to generated a metro map for a corpus it was not designed to be used on; international sports news. This demonstrated the need for common entities within a corpus to be identifiable without ambiguity, unlike for instance the names of international sports teams between different sporting disciplines. We did however find that the system generates useful maps for sports news corpora when restricted to individual sports, where there is less obvious name ambiguity, and the corpus itself suggests a schema for expected entities.

Section 4.3 ended with a more general note on the utility of a schema during the graph formation process, where metro lines are chosen from a pool of candidate keywords. Being able to select or reject lines based on a schematisation of expected entities within a corpus was not something we had considered during the implementation of the system, as it only came to light by using the system on data it was not designed for. Nevertheless, schema-based metro line selection and pruning is a novel concept which merits further investigation in future systems.

While accurate topic coverage and octilinear embeddings may be sufficient to evaluate the keyword extraction and map layout processes respectively, they do not provide any real insight into usability. That is to say, we may have designed a system which generates *good* metro maps both contextually and aesthetically, but the maps themselves have not been shown to have any effect on information overload. The next chapter therefore presents this evaluation in the form of two experiments.

Chapter 5
Empirical Evaluation

This chapter describes the process by which we designed and conducted a user study to evaluate our metro maps against unstructured RSS feed readers. Results and implications of this study are discussed, with full data available in Appendix C.4. The two metro maps used for the study have been included in Appendix C.3.

5.1 Scoping the Experiments

Typically, information retrieval tools can be evaluated in terms of accuracy and other quantifiable metrics against a canonical labelled dataset. Our system is less amenable to such evaluation due to the lack of existing datasets. The extensive user study conducted by Shahaf et al. [2012*b*] evaluated four distinct aspects of their metro map system; (1) accuracy of the algorithm's document selection in respect to a query; (2) user fact retrieval time; (3) user understanding at a macro level; and (4) task performance of users with a Metro Map versus an unstructured list.

For our system, the first aspect (accuracy of retrieval for queries) is not of interest, because the maps are not query-based or selected from a large fixed corpus with a reasonable likelihood of Type II errors. The second and third, while relevant to our system, do not effectively capture the spirit of its intended purpose. Our metro maps are not designed for search tasks; if a user has a specific question about an article in a metro map, they will most likely still have to open and read the article itself for the answer.

The maps the generated by the system exist to help the user form high-level connections between topics and to make sense of those they do not understand. A search task would therefore have to ask artificially simple questions in order for the answers to be derivable using the maps alone, and this would not be representative of typical news consumption tasks.

The final aspect of the Shahaf et al. study was therefore chosen as the focus of this evaluation. The goal of the experiments was to determine whether the structure of the metro maps alone provided a better overview to news content than a traditional RSS feed view. This question was answered by using Ho and Tang's [2001] dimensions of information overload to choose two perspectives to evaluate:

- Users' perceptions of **information quantity**.

  As information overload is based partially on readers' intrinsic estimations of quantity, we examined the effect that varying information format has on these estimations.

- The **contextual quantity** of our maps, as measured by task performance.

  We conducted an experiment similar to that of Pirolli et al. [1996], measuring the effect of varying information format on the breadth and depth of users' topic recollection.

5.2 Experimental Hypotheses

Considering perception and performance as our two separate measures of information overload, we developed the following hypotheses to be tested in two corresponding experiments:

Hypothesis 1 (5.1)**.** *Users' estimates of the number of articles in a metro map are lower than their estimates of the same number of articles when displayed in a list form.*

$$\begin{aligned} H_0(1) &: \mu E_{map} = \mu E_{list} \\ H_1(1) &: \mu E_{map} < \mu E_{list} \end{aligned} \tag{5.1}$$

Hypothesis 2 (5.2)**.** *Users recall a higher number of topics after using a metro map representation than after using the RSS feed list view.*

$$\begin{aligned} H_0(2) &: \mu T_{map} = \mu T_{list} \\ H_1(2) &: \mu T_{map} > \mu T_{list} \end{aligned} \tag{5.2}$$

Where μE_{map} and μE_{list} are the mean estimate values for the number of articles contained within a map and list respectively, and μT_{map} and μT_{list} are the mean number of topics successfully recalled.

5.3 Experimental Design

In order to increase the number of subjects participating in each condition and control for individual variances in ability, both experiments were conducted within-groups. With each subject participating in both conditions (*map* and *list*), two corpora of news articles were required. Subjects were therefore assigned to one of four groups, with counterbalancing of both the order in which the conditions were conducted, and the order in which the two corpora were read (Table 5.1.) Each group contained four participants.

	A, B	B, A
List, *Map*	(Group 1) *List* A **then** *Map* B	(Group 3) *List* B **then** *Map* A
Map, *List*	(Group 2) *Map* A **then** *List* B	(Group 4) *Map* B **then** *List* A

Table 5.1: Participant groups

5.3.0.2 Participants

In total, sixteen participants (four female, eight male, mean age = 21.8 years) enrolled in the study. All had perfect or corrected-to-perfect vision and hearing, and all were undergraduate students at the University of Bath. No participants were excluded on the basis of how much or how little news they consumed on a daily basis. Similarly, although it was recorded whether participants had previously used or currently use an RSS feed reader, this was not used as a selection criterion.

Before starting the experiments, the purpose of the evaluation was explained, and each participant signed a copy of the standard consent form in Appendix C.2. Afterwards, participants were debriefed and our hypotheses were explained to them.

5.4 Methodology

5.4.1 Estimate Task

The first experiment was designed to test Hypothesis 1. Participants were exposed to both modalities for one minute, after which they estimated the number of articles they had been looking at. To prevent conscious or subconscious counting by participants during the second condition, it was only after exposure to the second set of articles that subjects were asked to estimate the number of articles in each.

5.4.2 Index Task

The second experiment, which was designed to test Hypothesis 2, placed a significantly greater cognitive load on participants. This task is based on the evaluation of the Scatter/-Gather Browser by Pirolli et al. [1996], which required participants to draw a topic index, or *tree*, representing their understanding of topic structure within a corpus.

For each of the two conditions, participants first spent three minutes reading and exploring the news represented.[1] After the three minutes, the modality was minimised and the participants were given two minutes to write a topic index based on the news they had just read. An example topic index based on news in a different domain was shown before this task began to ensure participants knew what they were supposed to produce. Both datasets (and therefore all four conditions) contained 31 articles.

Finally, participants were asked to complete a Self-Assessment Manikin [Bradley and Lang, 1994] to rate the pleasure (labelled from 'unhappy' to 'happy'), arousal ('calm' to 'excited') and dominance ('not in control' to 'in control') they felt as a result of completing the index task. This process was then repeated with the second modality and data set, according to the participant's grouping. The counterbalanced ordering of two conditions between groups controlled for the practice effect in this task.

[1] As this task was completed after the Estimate Task (Section 5.4.1), the total time spent interacting with the data was actually four minutes.

5.4.3 Variables

The following section describes the independent variable in both studies, the dependent variables we measured, and the control variables which we either controlled through the design of the study or analysed as potential covariates.

5.4.3.1 Independent Variables

The independent variable in both experiments was the display modality; a nominal value which was either *metro map* or *list*. Participants completed the same tasks with both a metro map and a list modality, using a different news corpus for each.

5.4.3.2 Dependent Variables

- **Article Estimate (Hypothesis 1)**

  This is the raw estimate of the number of articles, and is measured on a ratio scale.

- **Primary topics (Hypothesis 2)**

  This is the number of 'top-level' topics identified in a participant's index which are present in the corpus, measured on a ratio scale. If participants identified a single primary topic (e.g. 'America'/ 'The USA'), this top level of the index was ignored and primary topics were counted as those in the next level.

- **Topic index depth (Hypothesis 2)**

  This is the maximum depth reached from any primary topic in a participant's index, measured on a ratio scale.

- **Total topics (Hypothesis 2)**

  This is the total number of topics identified at all levels in a participant's index which are present in the corpus, measured on a ratio scale.

- **Self-Assessment Manikin (SAM) score (Hypothesis 2)**

  The valence, arousal and dominance self-reported by participants after completing the Index Task. These were measured on the 9-point scale variant, as shown by the arousal scale in Figure 5.1.

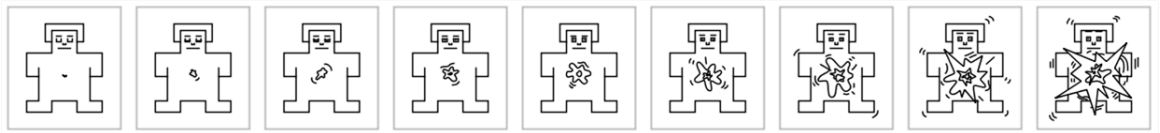


Figure 5.1: The 9-point SAM scale representing arousal, from *calm* to *excited* [Bradley and Lang, 1994]

5.4.3.3 Control Variables

The following are the participant characteristics and experimental conditions which were identified as potential covariates. Weekly news consumption, which we chose not to control through participant selection, was later examined as a influencing factor on performance.

- **Level of formal education**

  It was expected that differing levels of formal education would impact the relative performance of participants, however all participants were undertaking the final year of an undergraduate degree, so we did not account for this as a covariate.

- **Weekly news consumption time**

  It was expected that participants who consume more news on average would perform uniformly better across both conditions, but as subjects participated in both conditions, we do not expect to need to account for this as a covariate. This variable was measured on a discrete scale of hours per week.

- **Topic Familiarity**

  Since all subjects participated in both conditions, each subject completed the *metro map* condition with one set of articles and the *list* condition with the other set. The two corpora (A and B) were extracted from the same RSS feed[2] within the same week, to ensure background knowledge wouldn't confound performance between A and B.

5.5 Results and Discussion

A complete set of results for the following tests is available in Appendix C.4 (Hypothesis 1: Table C.4, Hypothesis 2: Table C.5, tests for normality: Table C.10, correlations: Tables C.7 - C.9), with raw data in Section C.5.

The use of one-tailed tests, even in the case of clear directional hypotheses, is controversial and may be seen as insufficiently conservative. However, for exploratory research like this which is necessarily limited in statistical power, we hope it is justified as exposing effects which seem promising. Future studies could test these same hypotheses more thoroughly.

5.5.1 Testing Hypothesis 1: Estimate Task

Our first hypothesis predicted that users' estimates for the number of articles represented on a map would be lower than their estimates of the same number of articles shown in the form of a list. A paired-samples t-test was conducted to compare estimates for the *map* and *list* conditions. Data was screened for violation of the following assumptions before the test was performed:

- **Continuous Measurement**: Both estimates were measured on a continuous scale.

[2]The Guardian, US News, week commencing 20/03/2017.

- **Independence**: Participants' estimates were independent, as the experiment placed them in isolation.
- **Outliers**: There were no outliers in either of the estimates.
- **Normality**: The normality assumption was tested with the Shapiro-Wilk test on *map estimate* (W=0.881, df=16, p=.080) and *list estimate* (W=0.954, df=16, p=.559). These results suggest normality is a reasonable assumption for the data.

Figure 5.2 shows the means for each condition split by corpus. Although the mean estimates for Corpus A were lower, this difference was not significant.

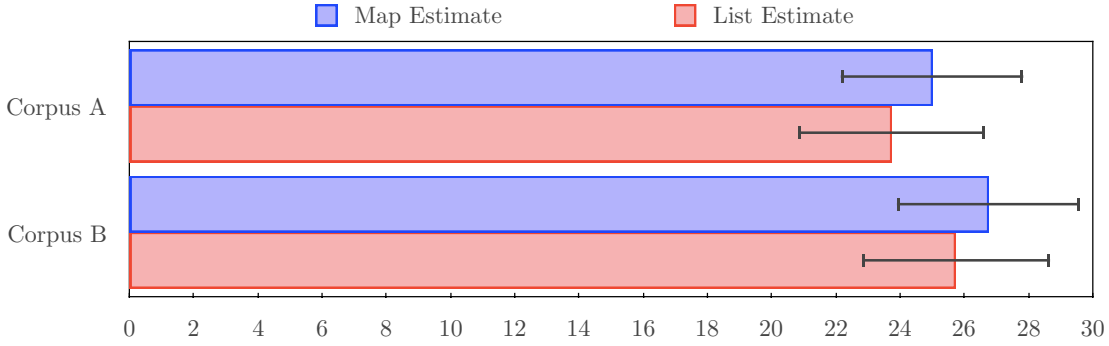


Figure 5.2: Comparison of *Map* and *List* estimates between the two corpora

The results of the t-test suggest there was no significant difference in estimates between the map (M=24.75, SD=7.188) and list (M=25.88, SD=6.820) conditions; $t(15)$=-0.504, p=.311 (one-tailed, $\alpha = .05$), providing insufficient evidence to reject H_0. We conclude that there is no significant difference between participants' perceptions of information quantity on the basis of modality alone.

5.5.2 Testing Hypothesis 2: Index Task

Our second hypothesis predicted that users would identify more topics in their index after using the map than after using the list. A second paired-samples t-test was conducted to compare the number of topics produced during the *map* and *list* conditions.

As in the previous test, there was no missing data, and data was screened for violation of measurement level, independence, outliers and normality (*map topics*: W=0.958, df=16, p=.633, *list topics*: W=0.917, df=16, p=.152) as before. Figure 5.3 verifies that there was no significant difference in total index topics produced between corpora A and B.

The results of the test did show a significant difference in total topics produced in the map (M=10.94, SD=4.389) and list (M=8.88, SD=2.156) conditions; $t(15)$=2.254, p=.20 (one-tailed, $\alpha = .05$), providing sufficient evidence to reject H_0. Since these results supported our hypothesis, we conducted a second round of tests in an attempt to determine whether the additional topics produced by users with metro maps was due to additional breadth or depth of their news recall. The additional dependent variables we measured during the index task were *primary topics* (topics at the highest level of the index, analogous to breadth of recall) and *topic depth* (the maximum depth reached in an index).

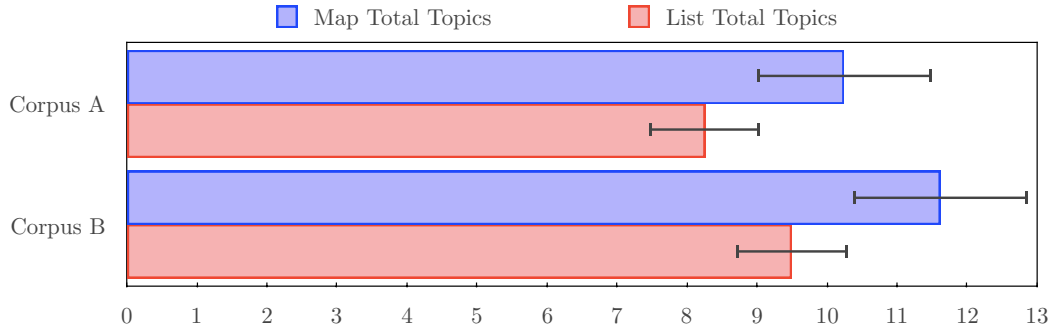


Figure 5.3: Comparison of *Map* and *List* total topics between the two corpora

Two further paired-samples t-tests were therefore conducted to compare the respective breadths and depths of participants' indexes between the two modalities (Figure 5.4).

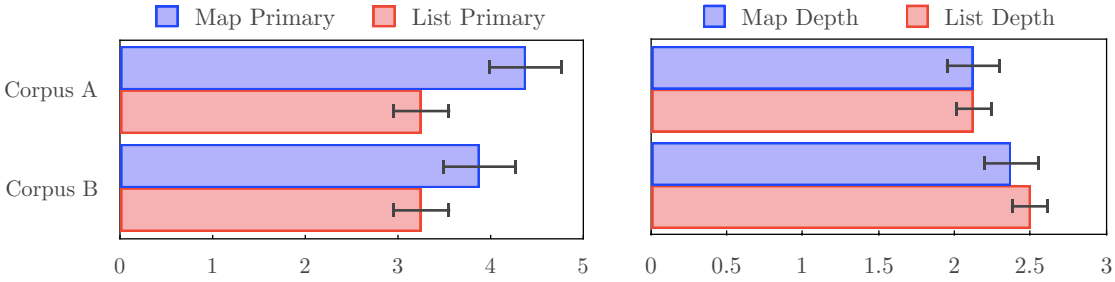


Figure 5.4: Comparison of *Map* and *List* topic breadth and depth between the two corpora

These tests showed no significant difference between map index depth (M=2.25, SD=0.577) and list index depth (M=2.31, SD=0.602); $t(15)$=-0.368, p=.359 (one-tailed, $\alpha = .05$), however, they did show a significant difference between map primary topics (M=4.19, SD=1.607) and list primary topics (M=3.25, SD=1.125) conditions; $t(15)$=1.996, p=.032, at the same significance level.

The implication of this result is that the additional topics participants were able to recall in the *map* condition were due to increased recall of the breadth of the news they had read, rather than increased in-depth recall of any particular topics.

5.5.3 The Effects of Weekly News Consumption

The average weekly news consumption of participants was a factor we identified as a potential influencer of task performance. Although our within-subjects design would have controlled any confounding effects, we analysed the Pearson correlation of this variable with several of our dependent variables to identify any unexpected correlations.

There was no significant correlation between average weekly news consumption time and participants' estimates of the number of articles on the map or in the list.

In the index task, news consumption was similarly correlated with total index topics for both the *map* (r=0.541, p=.031) and *list* (r=0.573, p=.020) conditions. As expected, high

news consumption positively influenced performance. The strongest correlation with news consumption however, was with the index depth in the *map* condition (r=0.606, p=.013). This, in contrast to the lack of any significant correlation in the *list* condition suggests that in drawing the users' focus to a wider breadth of topics, metro maps may sacrifice their understanding of topic depth.

5.5.4 Self-Assessment Manikin Scores

The three dimensions of affect measured by the Self-Assessment Manikin [Bradley and Lang, 1994] are pleasure (P), arousal (A), and dominance (D). Mehrabian [1991] classified every combination of high and low scores for each dimension, resulting in the following octants:

Exuberant	(+P, +A, +D)	vs.	*Bored*	(−P, −A, −D)
Relaxed	(+P, −A, +D)	vs.	*Anxious*	(−P, +A, −D)
Hostile	(−P, +A, +D)	vs.	*Docile*	(+P, −A, −D)
Distainful	(−P, −A, +D)	vs.	*Dependent*	(+P, +A, −D)

Although these were proposed as a taxonomy for human personality and temperaments, we expect the labels to generalise to emotional responses composed of the same three dimensions of affect, as measured using SAM scores.

In our analysis of the correlations between the three SAM dimensions, we found participants' pleasure to be to highly correlated to the levels of dominance they felt using the modality (*Map*: r=0.773, p <.001; *List*: r=0.685, p=.003) but uncorrelated with the level of arousal they reported. This suggests that enjoyment of both modalities is based on whether or not the participants felt in control while using them.

There was no correlation between the two modalities in terms of pleasure or dominance, but there was a correlation between *map* arousal and *list* arousal (r=0.646, p=.007), which suggests that arousal may be a function of enjoyment of news content itself and unrelated to the specific modality.

Due to the strong positive correlations between pleasure and dominance observed in SAM scores for both conditions, four of Mehrabian's octants are less relevant to our results. We can therefore restrict the set of typical emotional responses to the *map* and *list* conditions to tuples where sign(P) = sign(D):

Exuberant	(+P, +A, +D)	vs.	*Bored*	(−P, −A, −D)
Relaxed	(+P, −A, +D)	vs.	*Anxious*	(−P, +A, −D)

When pleasure and dominance are both high, it is not immediately obvious which emotional response is the desirable outcome between exuberance and relaxation. Taking each participant's exuberance as (P+A+D) and their relaxation as (P−A+D) and correlating these composite variables with the results of the index task, a strong link emerges between exuberance and total topics produced in the *map* condition (r=0.753, p=.001), as shown in Figure 5.5. No such correlation exists with relaxation in the *map* condition, or with either of these two variables in the *list* condition.

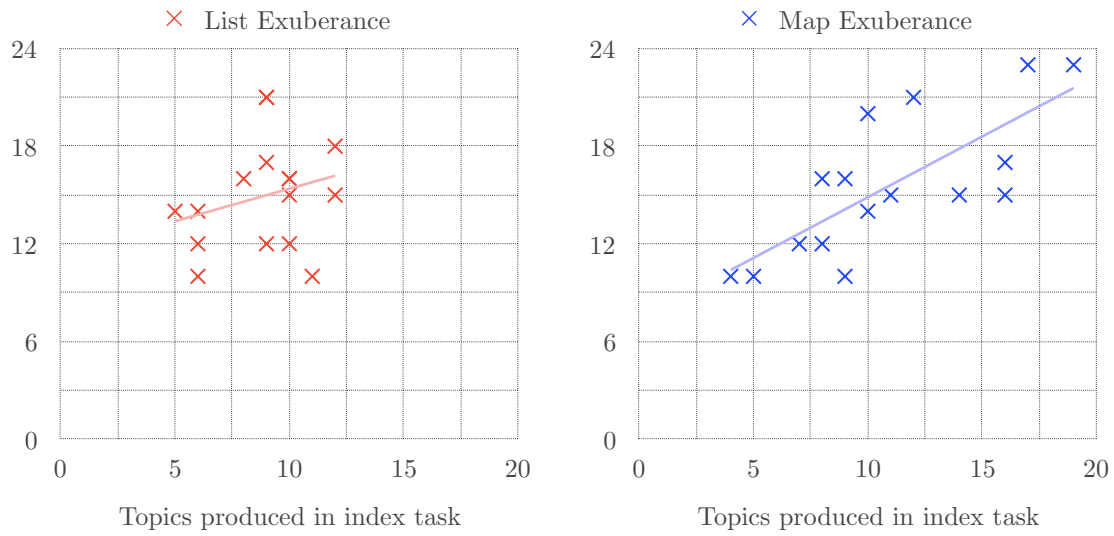


Figure 5.5: Comparison of correlations between index topics and exuberance

From the above correlations and the earlier results which support Hypothesis 2, it is apparent that the strength of the metro maps in terms of the dimensions of affect is that participants are more excited to use them, rather than more relaxed in doing so.

5.5.5 Participant Feedback

Participants' feedback on the metro maps was mixed, but with the majority of comments being positive. One participant, despite identifying more topics (both primary and total) using a metro map, reported lower happiness and dominance after using the map, stating "The list was easier. I could link the topics myself in a way that felt more natural. Using the map I didn't always agree with the categorisation."

This response suggests that there was a clash between the structure of the news as presented by the map and the participant's mental model of the underlying news, possibly exacerbated by detailed background knowledge, as the participant reported above average news consumption. We anticipate that if background knowledge does present a clash of mental models between user and metro map, then use of the *include* and *ignore* functionality by users more confident with current events news could help align the structure of the metro maps generated with their existing mental models.

In contrast to the first response, a different participant (whose scores were similar to the participant above during both conditions) said "It was easier to find stuff I cared about [using the map] with nothing in the way." This comment was in reference to the fact that if users wished to, they could choose to follow the stories along a single metro line and ignore everything else on the map, whereas with the list there was no filtering mechanism.

On the basis of our observations, participants seemed to remain more focused during the map condition than the list condition. This could be because the metro maps were more engaging as a modality, or that they required more concentration in order to explore, or simply because the concept was novel. One participant who expressed enjoyment of the map

condition, which she completed first, was repeatedly distracted during the list condition. While browsing the list, she commented "Sorry, I just completely disengage from politics." There were no similar expressions of frustration from this participant or any others during the map condition.

During the index task for the list condition, another participant asked whether topics could be repeated, "Because this one relates to both of those [parent topics]". Shortly after, he remarked "This really does suit being on a graph, doesn't it." This was perhaps a reflection on the demographic of our participants; 15 out of the 16 were Computer Science students, who are required to study elementary graph theory as part of their course.

The differences between these comments show the wide variation in responses to the system, even among the 16 students who participated in the experiments. Without conducting a larger survey into usability among a wider demographic of participants, it would be impossible to formulate a representative opinion on the system or its applications.

5.6 Summary

This chapter documented the evaluation the metro maps produced by the system in terms of how they affected users' perceptions of information quality, and to what degree they aided users in understanding the context of the news they read.

We found no significant evidence to support our first hypothesis; that users would estimate metro maps contained fewer articles than equivalent list views with the same number of articles. However, we did find evidence which supported our second hypothesis; that users recall a higher number of topics after using the metro maps than after using the list representation; $t(15)=2.254$, $p=.020$ (one-tailed, $\alpha = .05$). Recall in this task was measured by asking participants to produce a topic index, in the same style as the evaluation of the Scatter/Gather browser by Pirolli et al. [1996]. The total topics, index depth and index breadth of this index were counted.

Further examination of the topic indexes produced by participants lead to the conclusion that the larger indexes produced in the metro map condition could be attributed to due to participants recalling a greater breadth of topics, rather than recalling topics in any greater detail; $t(15)=1.996$, $p=.032$.

A discussion of the emotional responses to both modalities followed, where we generalised Mehrabian's [1991] temperament octants to affective interactions and using these to examine the correlations within participants' Self-Assessment Manikins [Bradley and Lang, 1994]. From this, we concluded that the level pleasure participants experience using either modality is highly correlated with whether or not they perceived themselves as being in control (*Map*: $r=0.773$, $p<.001$; *List*: $r=0.685$, $p=.003$). Arousal was identified as an independent factor in both modalities and is correlated between the two; it appears to be influenced by factors outside the scope of our experiments.

The chapter concluded by discussing some of the informal feedback participants gave during and after the study, illustrating somewhat polarised opinions on the utility of the maps.

Chapter 6
Conclusions

The following chapter outlines the contributions of this work and the limitations of both our system and evaluation. It also describes extensions which could be developed given additional time, and areas which would benefit from further research.

6.1 Summary of Contributions

This dissertation introduced the first application of the metro map metaphor [Shahaf et al., 2012*b*] to news corpora extracted from user-specified RSS feeds, in order to automatically generate structured topic maps based on current events news. These metro maps serve as an effective comprehension aid to young people in our current digital landscape of news overload; a claim which we evaluated empirically through a user study which directly compared our system with a typical RSS feed reader.

Our implementation includes a novel keyword extraction algorithm which performs keyword boosting using a knowledge base for entity disambiguation, and is tuned specifically for extracting entities from current events news. In the subdomain of metro map topology, our method also formalises *Line Coverage* and *Affinity* as an extension to the characteristics defined by Shahaf et al. [2012*b*], which can be used to compare the quality of candidate lines during the selection and pruning processes.

Due to the lack of any existing library for positioning or drawing metro maps, further contributions made include recommendations on tuning specific D3.js parameters within a force-directed layout to generate viable starting positions for stations on a metro map, and notes on the implementation of functions to calculate Stott's [2008] line straightness and octilinearity criteria in JavaScript. To the best of our knowledge, this is the first application of Stott's aesthetic criteria for metro map layout optimisation to non-geospatial maps.

The final contribution of this work is an empirical evaluation of the task performance of young adults using the system. The results of our experiments support our hypothesis that users recall more news topics from a news corpus using our metro maps than after using an unstructured RSS feed reader, and therefore substantiate our recommendations that future efforts to support the contextual linking of news articles should look to visualisation and particularly cartography for direction.

6.2 Limitations

The main limitation of this work is the lack of a formal deterministic algorithm for drawing metro maps, using D3 or otherwise. The approach presented in Section 3.5 is nondeterministic by necessity in order to generate initial starting positions which result in few or no edge crossings, but this process is non-convergent and often results in suboptimal embeddings, the worst cases of which are unusable.

As a result of this limitation, a significant effort was made to decouple the map drawing process from the other components of the system, such that it could be improved or replaced in future with the only changes required being to the interface at the serialisation stage. While the problem of generating planar embeddings for metro maps is known to be NP-hard, the maps drawn by our system are typically significantly smaller than metro maps which represent real existing transit networks, meaning it would not be unrealistic to apply a non-polynomial time algorithm.

A second important limitation of the system is the inability of the graph formation logic to recognise an overly sparse, overly connected, or overly populated map. All three of these properties are easily addressed in theory; an overly sparse graph should trigger a repeat of the keyword extraction process with a larger keyword vector per article, an overly connected graph should trigger a repeat of the same process with a smaller keyword vector per article, and an overly populated map should use both a smaller keyword vector and have its lowest ranking metro lines removed. However, in the given time, it was not possible to implement this logic in the system, resulting in a need to manually tune parameters such as the above during its operation.

In terms of experimental design, the weaknesses of this work lie in its limited scope. While our user study managed to capture a wide range of news consumption behaviours and attitudes among participants, all participants were students aged 20-23 who were technically proficient. 15 out of the 16 were Computer Science undergraduates, who are typically more familiar with graph theory and therefore more confident interpreting representations of abstract graphs than the general young adult population. This is a limitation which could be addressed by further experiments, as although the system was specifically conceptualised for use by young adults as a result of The Associated Press and Context-Based Research Group's [2008] findings, it would have strengthened our conclusions to have evaluated the system in a larger study covering a wider age bracket, and including participants from a non-technical background.

In addition to the lack of participant academic diversity, the scope of our measurement of task performance could also have been extended. The study we conducted was focussed exclusively on recall, but recall is only one of the six factors identified by Eppler [2004] as being strengthened by the use of visual metaphors; the others being motivation, the formation of new perspectives, support for learning, focusing of attention, and structuring of communication. It could be argued that the act of creating a topic index also evaluated the structure of participants' communication, but this still leaves four factors remaining which we did not evaluate. If our study had been more long-term, we could also have evaluated whether the daily use of metro maps supported participants' learning or increased their motivation to read the news, as these were the two factors identified in Chapter 1 as the most critical influencers of news fatigue.

6.3 Directions for Future Research

Throughout this work, we have touched on several extensions and adaptations which we envisioned for the system. In summary, these are as follows:

- A continually expanding corpus of past articles which can be linked by a contextual knowledge base such as Wikipedia, to extend newly generated metro lines back in time. This would provide context historical to unfolding events, but it would require a significant overhaul to the system, which currently cannot generate metro maps which integrate new content with previously processed articles.
- A sophisticated process for performing keyword extraction, again using a contextual knowledge base to disambiguate and identify entities within articles. This linkage could be preserved in the final visualisation, so that entities within summaries in the generated maps contain hyperlinks to their Wikipedia entries. Previous work using Wikipedia for these tasks which we consider to be relevant to this extension includes [Milne and Witten, 2008] and Mihalcea and Csomai's *Wikify* [2007].
- The application of the [Stott et al., 2011, 2008] multicriteria optimisation function to the metro maps generated by our system. This would require either a JavaScript or Python implementation of the algorithm, which could be optimised by removing all criteria related to station labelling.

However, during the design and implementation of the system, the initial stage of our pipeline was the stage which was the most rapidly developed and the least explored in terms of future potential. Therefore, the remainder of this chapter outlines topics we consider to be of interest for future work within this first stage of operation; the acquisition of the news content itself.

6.3.1 Real-Time News Tracking for Automated Collation

One short-term goal we suggest is to develop the system to a stage where no manual tuning is required to prune the candidate metro lines to a level which strikes a balance between useful and usable. After this, article collation would a simple process to automate, meaning maps could be automatically generated on a periodic basis with no need for user input.

Web services such as News API[1], which publishes a continual stream of JSON metadata containing live headlines from 72 major worldwide news publishers, are gaining traction as the demand for immediate information increases. Although metro maps have not been designed with live updating in mind, the use of services such as News API could serve as a replacement for specific RSS feeds, removing the major initial parameter of the system.

The key questions posed by the concept of real-time metro maps of news are related to the mechanics of updating, and how stateful the updates should be. Should the publishing of five new headlines trigger a complete regeneration of the map (which would potentially result in a new set of metro lines) or should metro lines be fixed over a specific period of

[1] `https://newsapi.org`

time? Would users want to be able to 'pin' metro lines to indicate that the system should consider this an ongoing topic of interest and always choose it as a candidate?

We suspect that there is a limitation on the usefulness of metro maps when their data is continually updating in real-time due to the structural changes which could occur within the maps, but the core idea of dispensing with specific RSS feeds and simply watching major news outlets for trends is an interesting one. While users may find it beneficial to combatting news fatigue to specify exactly where they want their news to originate from or which topics they want to read about, real-time headlines also take the effort out of specifying feeds for users who are indifferent about the sources of the news they consume.

6.3.2 Social Media, and the Dangers of the Echo Chamber

In the current information landscape, it is impossible to discuss real-time content publishing without discussing the impact social networks have on news consumption. The evolution of social networking websites – in particular, Twitter – has been instrumental in the erosion of the dichotomy between news media and social media.

The Pew Research Centre reported in 2015 that 61% of millenials and 51% of Generation X rank Facebook as their primary source for political news [Mitchell et al., 2015]. A risk of this kind of news consumption is that Facebook's newsfeed algorithms determine which news to show users based on their personal preferences and past engagement with articles [Tufekci, 2015]. Although Facebook and Twitter are often singled out for heavy criticism due to how they filter news content during political campaigns, social media is not entirely to blame; other personal news recommendation systems work in the same manner, albeit more overtly.

The echo chambers of content which arise from over-accounting for user preferences in news personalisation were coined the *filter bubble* by Pariser [2012], and they result in a deliberate suppression of content which challenges our existing views. A study funded by Facebook itself confirmed this phenomenon, but attributed the filter bubble effect to the promotion of content posted by users' networks of close friends, who often share similar political views [Bakshy et al., 2015].

> *"The new generation of Internet filters looks at the things you seem to like - the actual things you've done, or the things people like you like - and tries to extrapolate. They are prediction engines, constantly creating and refining a theory of who you are and what you'll do and want next. Together, these engines create a unique universe of information for each of us - what I've come to call a filter bubble - which fundamentally alters the way we encounter ideas and information."* [Pariser, 2012, p.29]

While our system was developed to run on user-specified feeds, we argue that this kind of explicit content selection is less harmful than the implicit selection which occurs within the filter bubble. As Pariser states, two of the most harmful dynamics of the filter bubble are that we don't choose to enter it, and that we don't always realise we're inside it at all. By forcing users to explicitly declare their trusted sources of information, we hope that most users would select content which Facebook's and Twitter's algorithms would deem to be

outside of the narrow scope of their filter bubbles. We also expect that users would be less likely to seek out websites known for producing clickbait articles of their own accord [Chen et al., 2015]. However, this claim is unverified, and future work is needed to investigate the impact that blindly filtered content recommendations can have on young people. Even in spite of this lack of evidence, we are reluctant to encourage any further work on our system which is based on personalised news recommendations, especially if such recommendations are done without explicitly notifying users.

In our attempts to address news overload, we cannot compromise on the exposure of young adults in particular to a diverse and balanced range of news content. To do otherwise would be a regression, regardless of innovation behind the approach.

6.3.2.1 News Verification

A second phenomenon which has been blamed on the filter bubble effect is the recent unprecedented rise of *fake news*, that is, deliberately misleading news content which is portrayed as authentic. The internet has always contained misinformation, but it has never been as accessible as it is today. The echo chambers cultivated by social media platforms have been blamed for encouraging speculative rumours based on "confusion about causation" [Del Vicario et al., 2016, p.1].

Although in late 2016 both Facebook and Google took public action against the spread of misinformation by removing websites known for publishing misleading content from their advertising networks, significant damage has already been done. Any content which conflicts with an opposing political ideology can now be dismissed as fake news, even when the content is genuine. Additionally, while Facebook has now partnered with Snopes to develop fake-news-detection software as part of its platform, it relies on users manually flagging suspicious content as potentially fake [DiFranzo and Gloria-Garcia, 2017], meaning misleading content can still reach users' feeds.

An opportunity presents itself here in the space between content publishers (legitimate or otherwise) and the news feeds of young adults on Twitter and Facebook. If a trusted aggregator were to publish news content on either of these platforms, users could know that they were both receiving content from a wide range of publishing outlets, and that the content itself was genuine.

Verifiable, up-to-date visualisations which could be embedded in Facebook or Twitter posts would reduce the investment required by users who may otherwise not be willing to split their news consumption between social networks and other services. It is in the bridging of this gap between trusted sources and the social networks where young people consume the majority of their news that we see great potential for the development of NewsGraph and other such platforms.

Bibliography

Associated Press and Context-Based Research Group [2008], 'A new model for news: Studying the deep structure of young adult news consumption'. Accessed: 26/10/2016.
URL: *http://manuscritdepot.com/edition/documents-pdf/newmodel.pdf*

Bakshy, E., Messing, S. and Adamic, L. A. [2015], 'Exposure to ideologically diverse news and opinion on facebook', *Science* **348**(6239), 1130–1132.

Barzilay, R., McKeown, K. R. and Elhadad, M. [1999], Information fusion in the context of multi-document summarization, *in* 'Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics', Association for Computational Linguistics, pp. 550–557.

Bergamaschi, S., Guerra, F. and Leiba, B. [2010], 'Guest editors' introduction: information overload', *IEEE Internet Computing* **14**(6), 10–13.

Binh Tran, G. [2013], Structured summarization for news events, *in* 'Proceedings of the 22nd International Conference on World Wide Web', WWW '13 Companion, ACM, New York, NY, USA, pp. 343–348.
URL: *http://doi.acm.org/10.1145/2487788.2487940*

Bradley, M. M. and Lang, P. J. [1994], 'Measuring emotion: the self-assessment manikin and the semantic differential', *Journal of behavior therapy and experimental psychiatry* **25**(1), 49–59.

British Broadcasting Corporation [2013], 'Tube 150th anniversary: How the underground map evolved'. Accessed: 4/1/2017.
URL: *http://www.bbc.co.uk/news/uk-england-london-20943525*

Bruns, A., Highfield, T. and Lind, R. A. [2012], 'Blogs, twitter, and breaking news: The produsage of citizen journalism', *Produsing theory in a digital world: The intersection of audiences and production in contemporary theory* **80**(2012), 15–32.

Bun, K. K. and Ishizuka, M. [2002], 'Topic extraction from news archive using tf-pdf algorithm', *Proceedings of the 3rd International Conference on Web Information Systems Engineering* .

Burkhard, R. A. [2004], Learning from architects: the difference between knowledge visualization and information visualization, *in* 'Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on', IEEE, pp. 519–524.

Carpena, P., Bernaola-Galván, P., Hackenberg, M., Coronado, A. and Oliver, J. [2009], 'Level statistics of words: Finding keywords in literary texts and symbolic sequences', *Physical Review E* **79**(3), 035102.

Chen, Y., Conroy, N. J. and Rubin, V. L. [2015], Misleading online content: Recognizing clickbait as "false news", *in* 'Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection', WMDD '15, ACM, New York, NY, USA, pp. 15–19.
URL: *http://doi.acm.org/10.1145/2823465.2823467*

Del Vicario, M., Bessi, A., Zollo, F., Petroni, F., Scala, A., Caldarelli, G., Stanley, H. E. and Quattrociocchi, W. [2016], 'The spreading of misinformation online', *Proceedings of the National Academy of Sciences* **113**(3), 554–559.

DiFranzo, D. and Gloria-Garcia, K. [2017], 'Filter bubbles and fake news', *XRDS* **23**(3), 32–35.
URL: *http://doi.acm.org/10.1145/3055153*

Dredze, M., McNamee, P., Rao, D., Gerber, A. and Finin, T. [2010], Entity disambiguation for knowledge base population, *in* 'Proceedings of the 23rd International Conference on Computational Linguistics', COLING '10, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 277–285.
URL: *http://dl.acm.org/citation.cfm?id=1873781.1873813*

Dwyer, T. [2009], Scalable, versatile and simple constrained graph layout, *in* 'Computer Graphics Forum', Vol. 28, Wiley Online Library, pp. 991–998.

Eppler, M. J. [2004], Visuelle kommunikationder einsatz von graphischen metaphern zur optimierung des wissenstransfers, *in* 'Wissenskommunikation in Organisationen', Springer, pp. 13–31.

Eppler, M. J. [2006], 'A comparison between concept maps, mind maps, conceptual diagrams, and visual metaphors as complementary tools for knowledge construction and sharing', *Information visualization* **5**(3), 202–210.

Ferrari, D. and Mezzalira, L. [1970], 'A computer-aided approach to integrated circuit layout design', *Computer-Aided Design* **2**(2), 19–23.

Finkel, J. R. and Manning, C. D. [2009], Nested named entity recognition, *in* 'Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1', Association for Computational Linguistics, pp. 141–150.

Fischer, G. and Stevens, C. [1991], Information access in complex, poorly structured information spaces, *in* 'Proceedings of the SIGCHI Conference on Human Factors in Computing Systems', CHI '91, ACM, New York, NY, USA, pp. 63–70.

Garg, A. and Tamassia, R. [1994], On the computational complexity of upward and rectilinear planarity testing, *in* 'International Symposium on Graph Drawing', Springer, pp. 286–297.

@GoogleTrends Twitter account [2016], 'Tweet: *+250% spike in "what happens if we leave the EU" in the past hour*', `http://twitter.com/GoogleTrends/status/746137920940056578`. Accessed: 27/10/2016.

Goyal, R., Malla, R., Bagchi, A., Mehta, S. and Ramanath, M. [2013], Esthete: A news browsing system to visualize the context and evolution of news stories, *in* 'Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management',

CIKM '13, ACM, New York, NY, USA, pp. 2529–2532.
URL: *http://doi.acm.org/10.1145/2505515.2508208*

Hargreaves, I., Thomas, J., Commission, I. T. and Commission, G. B. B. S. [2002], *New news, old news: an ITC and BSC research publication*, Independent Television Commission.
URL: *https://books.google.co.uk/books?id=VZ-hPAAACAAJ*

Havre, S., Hetzler, E., Whitney, P. and Nowell, L. [2002], 'Themeriver: Visualizing thematic changes in large document collections', *IEEE transactions on visualization and computer graphics* **8**(1), 9–20.

Ho, J. and Tang, R. [2001], Towards an optimal resolution to information overload: An infomediary approach, *in* 'Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work', GROUP '01, ACM, New York, NY, USA, pp. 91–96.
URL: *http://doi.acm.org/10.1145/500286.500302*

Hochmair, H. [2009], 'The influence of map design on route choice from public transportation maps in urban areas', *The Cartographic Journal* **46**(3), 242–256.

Holton, A. E. and Chyi, H. I. [2012], 'News and the overloaded consumer: Factors influencing information overload among news consumers', *Cyberpsychology, Behavior, and Social Networking* **15**(11), 619–624.

Husin, H. S., Thom, J. A. and Zhang, X. [2014], Analysing user access to an online newspaper, *in* 'Proceedings of the 2014 Australasian Document Computing Symposium', ADCS '14, ACM, New York, NY, USA, pp. 77:77–77:80.

@jeffjarvis Twitter account [2016], 'Tweet: *Google knowledge graph has more than 70 billion facts about people, places, things. + language, image, voice translation*', `https://twitter.com/jeffjarvis/status/783338071316135936`. Accessed: 20/02/2017.

Kobourov, S. G. [2012], 'Spring embedders and force directed graph drawing algorithms', *CoRR* **abs/1201.3011**.
URL: *http://arxiv.org/abs/1201.3011*

Lin, C.-Y. and Hovy, E. [1997], Identifying topics by position, *in* 'Proceedings of the fifth conference on Applied natural language processing', Association for Computational Linguistics, pp. 283–290.

Lipton, R. J., North, S. C. and Sandberg, J. S. [1985], A method for drawing graphs, *in* 'Proceedings of the first annual symposium on Computational geometry', ACM, pp. 153–160.

Liu, B., Han, H., Noro, T. and Tokuda, T. [2009], Personal news rss feeds generation using existing news feeds, *in* 'International Conference on Web Engineering', Springer, pp. 419–433.

Liu, S., Zhou, M. X., Pan, S., Qian, W., Cai, W. and Lian, X. [2009], Interactive, topic-based visual text summarization and analysis, *in* 'Proceedings of the 18th ACM Conference on Information and Knowledge Management', CIKM '09, ACM, New York, NY, USA, pp. 543–552.

Mehrabian, A. [1991], Outline of a general emotion-based theory of temperament, *in* 'Explorations in Temperament', Springer, pp. 75–86.

Mihalcea, R. and Csomai, A. [2007], Wikify!: Linking documents to encyclopedic knowledge, *in* 'Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management', CIKM '07, ACM, New York, NY, USA, pp. 233–242.
URL: *http://doi.acm.org/10.1145/1321440.1321475*

Milne, D. and Witten, I. H. [2008], Learning to link with wikipedia, *in* 'Proceedings of the 17th ACM Conference on Information and Knowledge Management', CIKM '08, ACM, New York, NY, USA, pp. 509–518.
URL: *http://doi.acm.org/10.1145/1458082.1458150*

Mitchell, A., Gottfried, J. and Matsa, K. E. [2015], 'Facebook Top Source for Political News Among Millennials'.
URL: *http://www.journalism.org/2015/06/01/facebook-top-source-for-political-news-among-millennials*

Nallapati, R. [2003], Semantic language models for topic detection and tracking, *in* 'Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Proceedings of the HLT-NAACL 2003 student research workshop-Volume 3', Association for Computational Linguistics, pp. 1–6.

Nguyen, P. H., Xu, K., Walker, R. and Wong, B. [2014*a*], 'Timesets: timeline visualization for sensemaking'.

Nguyen, P. H., Xu, K., Walker, R. and Wong, B. W. [2014*b*], Schemaline: timeline visualization for sensemaking, *in* 'Information Visualisation (IV), 2014 18th International Conference on', IEEE, pp. 225–233.

Nöllenburg, M. and Wolff, A. [2005], Automated drawing of metro maps, PhD thesis, Technical University of Munich. Accessed: 15/12/2016.
URL: *https://www.semanticscholar.org/paper/Automated-Drawing-of-Metro-Maps-Nllenburg-Wolff*

Nordenson, B. [2008], 'Overload! Journalism's battle for relevance in an age of too much information', *Columbia Journalism Review* **47**(4), 30.

O'donnell, A. M., Dansereau, D. F. and Hall, R. H. [2002], 'Knowledge maps as scaffolds for cognitive processing', *Educational psychology review* **14**(1), 71–86.

Old, L. J. [2002], Information cartography: Using gis for visualizing nonspatial data, *in* 'Proceedings of the ESRI International Users Conference', Citeseer.

O'Shea, M. and Levene, M. [2011], 'Mining and visualising information from rss feeds: a case study', *International Journal of Web Information Systems* **7**(2), 105–129.

Palmer, D. D. [2000], Tokenisation and sentence segmentation, *in* 'Handbook of natural language processing', CRC Press.

Pariser, E. [2012], *The Filter Bubble: How the New Personalized Web Is Changing What We Read and How We Think*, Penguin Books, New York, NY, USA.

Pentina, I. and Tarafdar, M. [2014], 'From "information" to "knowing": Exploring the role of social media in contemporary news consumption', *Computers in Human Behavior* **35**, 211–223.

Pera, M. S. and Ng, Y.-K. [2008], 'Utilizing phrase-similarity measures for detecting and clustering informative rss news articles', *Integrated Computer-Aided Engineering* **15**(4), 331–350.

Phuvipadawat, S. and Murata, T. [2010], Breaking news detection and tracking in twitter, *in* 'Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on', Vol. 3, IEEE, pp. 120–123.

Pirolli, P., Schank, P., Hearst, M. and Diehl, C. [1996], Scatter/gather browsing communicates the topic structure of a very large text collection, *in* 'Proceedings of the SIGCHI conference on Human factors in computing systems', ACM, pp. 213–220.

Purcell, K., Rainie, L., Mitchell, A., Rosenstiel, T. and Olmstead, K. [2010], 'Understanding the participatory news consumer', *Pew Internet and American Life Project* **1**, 19–21.

Purchase, H. [1997], Which aesthetic has the greatest effect on human understanding?, *in* 'International Symposium on Graph Drawing', Springer, pp. 248–261.

Purchase, H. C., Cohen, R. F. and James, M. I. [1997], 'An experimental study of the basis for graph drawing algorithms', *J. Exp. Algorithmics* **2**.
URL: *http://doi.acm.org/10.1145/264216.264222*

Rennison, E. [1994], Galaxy of news: An approach to visualizing and understanding expansive news landscapes, *in* 'Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology', UIST '94, ACM, New York, NY, USA, pp. 3–12.
URL: *http://doi.acm.org/10.1145/192426.192429*

Rosen, J. [2008], 'National explainer: A job for journalists on the demand side of news', `http://archive.pressthink.org/2008/08/13/national_explain.html`. Accessed: 2/11/2016.

Russell, D. M., Slaney, M., Qu, Y. and Houston, M. [2006], Being literate with large document collections: Observational studies and cost structure tradeoffs, *in* 'Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)', Vol. 3, IEEE, pp. 55–55.

Salton, G. and Buckley, C. [1988], 'Term-weighting approaches in automatic text retrieval', *Information processing & management* **24**(5), 513–523.

Sayyadi, H., Hurst, M. and Maykov, A. [2009], Event detection and tracking in social streams., *in* 'Icwsm'.

Schick, A. G., Gordon, L. A. and Haka, S. [1990], 'Information overload: A temporal approach', *Accounting, Organizations and Society* **15**(3), 199–220.

Shahaf, D. and Guestrin, C. [2010], Connecting the dots between news articles, *in* 'Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 623–632.

Shahaf, D., Guestrin, C. and Horvitz, E. [2012*a*], Metro maps of science, *in* 'Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '12, ACM, New York, NY, USA, pp. 1122–1130.
URL: *http://doi.acm.org/10.1145/2339530.2339706*

Shahaf, D., Guestrin, C. and Horvitz, E. [2012*b*], Trains of thought: Generating information maps, *in* 'Proceedings of the 21st International Conference on World Wide Web', WWW '12, ACM, New York, NY, USA, pp. 899–908.
URL: *http://doi.acm.org/10.1145/2187836.2187957*

Shahaf, D., Guestrin, C., Horvitz, E. and Leskovec, J. [2015], 'Information cartography', *Commun. ACM* **58**(11), 62–73.
URL: *http://doi.acm.org/10.1145/2735624*

Shahaf, D., Yang, J., Suen, C., Jacobs, J., Wang, H. and Leskovec, J. [2013], Information cartography: creating zoomable, large-scale maps of information, *in* 'Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 1097–1105.

Shneiderman, B. [1996], The eyes have it: A task by data type taxonomy for information visualizations, *in* 'Visual Languages, 1996. Proceedings., IEEE Symposium on', IEEE, pp. 336–343.

Singh, J., Anand, A., Setty, V. and Anand, A. [2015], Exploring long running news stories using wikipedia, *in* 'Proceedings of the ACM Web Science Conference', WebSci '15, ACM, New York, NY, USA, pp. 57:1–57:2.
URL: *http://doi.acm.org/10.1145/2786451.2786489*

Skupin, A. [2000], From metaphor to method: Cartographic perspectives on information visualization, *in* 'Proceedings of the IEEE Symposium on Information Vizualization 2000', INFOVIS '00, IEEE Computer Society, Washington, DC, USA, pp. 91–.
URL: *http://dl.acm.org/citation.cfm?id=857190.857692*

Sparck Jones, K. [1972], 'A statistical interpretation of term specificity and its application in retrieval', *Journal of documentation* **28**(1), 11–21.

Stott, J. [2008], Automatic layout of metro maps using multicriteria optimisation, PhD thesis, University of Kent. Accessed: 11/11/2016.
URL: *http://www.jstott.me.uk/thesis/thesis-final.pdf*

Stott, J., Rodgers, P., Martinez-Ovando, J. C. and Walker, S. G. [2011], 'Automatic metro map layout using multicriteria optimization', *IEEE Transactions on Visualization and Computer Graphics* **17**(1), 101–114.

Strong, D. M., Lee, Y. W. and Wang, R. Y. [1997], 'Data quality in context', *Communications of the ACM* **40**(5), 103–110.

Tamassia, R. [1987], 'On embedding a graph in the grid with the minimum number of bends', *SIAM Journal on Computing* **16**(3), 421–444.

Teitler, B. E., Lieberman, M. D., Panozzo, D., Sankaranarayanan, J., Samet, H. and Sperling, J. [2008], Newsstand: a new view on news, *in* 'Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems', ACM, p. 18.

Transport For London [2014], 'Harry Beck's Tube map', `https://tfl.gov.uk/corporate/about-tfl/culture-and-heritage/art-and-design/harry-becks-tube-map`. Accessed: 18/1/2017.

Tufekci, Z. [2015], 'How Facebook's algorithm suppresses content diversity (modestly) and how the newsfeed rules the clicks'.
URL: *https://medium.com/message/how-facebook-s-algorithm-suppresses-content-diversity-modestly-how-the-newsfeed-rules-the-clicks-b5f8a4bb7bab*

Wang, T., Yu, N., Li, Z. and Li, M. [2006], nReader: Reading News Quickly, Deeply and Vividly, *in* 'CHI '06 Extended Abstracts on Human Factors in Computing Systems', CHI EA '06, ACM, New York, NY, USA, pp. 1385–1390.

Waters, K. [2009], 'Prioritization using moscow', *Agile Planning* **12**.

Weick, K. E., Sutcliffe, K. M. and Obstfeld, D. [2005], 'Organizing and the process of sensemaking', *Organization science* **16**(4), 409–421.

Wise, J. A., Thomas, J. J., Pennock, K., Lantrip, D., Pottier, M., Schur, A. and Crow, V. [1995], Visualizing the non-visual: Spatial analysis and interaction with information from text documents, *in* 'Proceedings of the 1995 IEEE Symposium on Information Visualization', INFOVIS '95, IEEE Computer Society, Washington, DC, USA, pp. 51–.

Yen, J.-C., Lee, C.-Y., Chen, I. et al. [2012], 'The effects of image-based concept mapping on the learning outcomes and cognitive processes of mobile learners', *British Journal of Educational Technology* **43**(2), 307–320.

Yi, J. S., Kang, Y.-a., Stasko, J. T. and Jacko, J. A. [2008], Understanding and characterizing insights: How do people gain insights using information visualization?, *in* 'Proceedings of the 2008 Workshop on BEyond Time and Errors: Novel evaLuation Methods for Information Visualization', BELIV '08, ACM, New York, NY, USA, pp. 4:1–4:6.
URL: *http://doi.acm.org/10.1145/1377966.1377971*

Ziemkiewicz, C. and Kosara, R. [2009], Preconceptions and individual differences in understanding visual metaphors, *in* 'Computer Graphics Forum', Vol. 28, Wiley Online Library, pp. 911–918.

Appendix A
Full Requirements Specification

A.1 Functional Requirements

1 Article Retrieval

1.1 Description

The article retrieval component of the system will accept the URL of one or more RSS feeds, collect links to the articles referenced and parse the text of those articles for further processing.

1.2 Functional Requirements

F1.1 The system must accept any standard XML document compliant with the RSS 2.0 specification[1], i.e. it should not be specific to any particular news provider

F1.2 The system must be able to extract a specified number of articles from an RSS feed, in reverse chronological order.

F1.3 The system must be able to download the textual content and/or raw HTML for each article.

F1.4 The system should accept multiple RSS feeds from one or more news provider(s) and merge their content into one collection.

F1.5 The system could verify new article URLs against the URLs of imported articles to ensure no articles are duplicated.

2 Keyword Extraction

2.1 Description

The keyword extraction stage will tokenise the parsed article text and determine a set of significant keywords for each article individually.

[1] http://cyber.harvard.edu/rss/rss.html

2.2 Functional Requirements

F2.1 The system must tokenise articles in order to perform basic natural language processing such as stop-word extraction and lemmatising.

F2.2 The system must implement a method for keyword extraction, and calculate a corresponding measure of relative importance for each keyword such as tf-idf.

F2.3 The system could attempt to combine keywords it considers equivalent (e.g. *UK* and *United Kingdom*) to form stronger keyword matches between or within articles.

F2.4 The system could use external services (e.g. Google's Knowledge Graph API[2]) to query any extracted keywords, in order to gain further insight or perform entity disambiguation [Dredze et al., 2010].

3 Graph Building

3.1 Description

This process involves determining a set of corpus keywords from the union of all the articles' keywords to form connected paths of edges (*lines*), and fitting a maximal number of articles into the resulting graph.

3.2 Functional Requirements

F3.1 The system must analyse the keywords extracted from all articles in a corpus to choose a set of the n most significant topics, where n is either predetermined or user-specified.

F3.2 The system must use the extracted topics and the publish dates (which form a natural ordering of nodes) of the articles to form a directed graph, with articles as vertices and common topic storylines as edges.

F3.3 The system should accept a user-specified topic or list of topics to include or exclude from the graph.

F3.4 The system should choose topics which are specific to some but not all articles in the collection, so as to avoid highly correlated topic keywords.

F3.5 The system could support exporting generated graphs in a graph description language e.g. DOT[3] or GraphML[4].

F3.6 The system could attempt to combine keywords to form topics if it considers them highly correlated.

F3.7 The system could attempt to maximise the coverage of the topic selection, i.e. maximise the number of articles covered by a given set of keywords.

[2] http://www.google.com/intl/es419/insidesearch/features/search/knowledge.html
[3] http://www.graphviz.org/content/dot-language
[4] http://graphml.graphdrawing.org

4 Map Drawing

4.1 Description

The visualisation component will generate an interactive visualisation of the graph which can be used to explore the corpus as a whole and drill-down to the individual article level.

4.2 Functional Requirements

F4.1 The system must provide the capability for users to visualise the graph structures it generates using any HTML5 compliant web browser.

F4.2 Metro lines must each be drawn in a different colour which contrasts that of other lines.

F4.3 The visualisation must include a key mapping metro line names to their colours on the map.

F4.4 Stations must be drawn with common (non-unique) symbols, with a distinction made for interchange stations.

F4.5 The system must provide drill-down details for nodes, e.g. by providing a hyperlink to the original article or embedding static content from each article within the visualisation itself to provide a preview.

F4.6 The maps generated should be readable by ensuring nodes and edges do not overlap with each other where possible.

F4.7 In support of F4.6, to improve the readability of the map, nodes should not be labelled with article titles.

F4.8 The maps generated should adhere (where possible) to the metro map aesthetics defined in [Stott, 2008, Stott et al., 2011] to preserve the familiarity of the metaphor.

F4.9 The system could allow some degree of interactive customisation which does not change the underlying structure of the graph, such as dragging nodes or changing attributes including colour.

5 Storage and Persistence

5.1 Description

This component of the system is responsible for saving and importing previously downloaded corpora and reconstructing their graphs.

5.2 Functional Requirements

F5.1 The system must support the importing/exporting of graph and article data in an intermediate data form, in order to fully reconstruct graphs it had previously created.

F5.2 By default, the articles collected by each run of the system must be treated as a new corpus so keyword ranking is deterministic for any given feed.

A.2 Nonfunctional Requirements

1 Security

The system will not require any kind of authentication to use, and will only stores data which is publicly available. As there are no security regulations which govern its usage, security is not a critical consideration and there are only two associated requirements.

NF1.1 The system will not collect any data during installation and usage without obtaining consent from the user.

NF1.2 The system will not transmit any data which was necessary to collect or generate, including log files, without obtaining explicit consent from the user.

2 Software Quality

The following list specifies the system's core requirements in terms of portability, source control, testability, usability and documentation. There are no specific performance metric requirements for the system at this stage of its development.

NF2.1 The system must not use any platform-specific libraries, functions or commands.

NF2.2 The system must provide a `requirements.txt`[5] file or similar, to allow its dependencies to be installed using Pip.

NF2.3 The system must be versioned and privately hosted on GitHub.

NF2.4 The implementation of the system should include a severity-based logging facility which writes to a text file, for use during debugging and testing.

NF2.5 The system must provide a non-interactive help facility for users.

NF2.6 The system should provide visual feedback during computationally expensive tasks to show task progress, e.g. with loading bars.

[5] https://pip.readthedocs.io/en/1.1/requirements.html

Appendix B
Implementation Raw Data

B.1 Token/Entity Data

Table B.1: Entities as a percentage of total tokens from 40 BBC Politics articles.

Tokens	Entities	Entities (%)
81	7	8.64198
129	8	6.20155
167	9	5.38922
181	15	8.28729
252	17	6.74603
264	15	5.68182
272	11	4.04412
275	17	6.18182
291	21	7.21649
343	22	6.41399
349	14	4.01146
366	23	6.28415
367	28	7.62943
367	14	3.81471
404	42	10.39604
409	40	9.77995
413	41	9.92736
434	25	5.76037
444	20	4.5045
453	29	6.40177
480	28	5.83333
498	41	8.23293
503	30	5.96421
523	35	6.69216
527	17	3.22581
536	62	11.56716
562	39	6.9395
562	32	5.69395
571	42	7.35552
618	33	5.33981

Table B.1: Entities as a percentage of total tokens from 40 BBC Politics articles.

Tokens	Entities	Entities (%)
621	38	6.11916
674	35	5.19288
688	34	4.94186
710	63	8.87324
722	65	9.00277
831	76	9.14561
920	51	5.54348
987	83	8.40932
1119	52	4.64701
1243	52	4.18343

Appendix C
Empirical Evaluation Data

C.1 Department of Computer Science Ethics Checklist

UNIVERSITY OF BATH

Date: 16/03/17

This document describes the 13 issues that need to be considered carefully before students or staff involve other people ("participants") for the collection of information as part of their project or research.

Q.1 **Have you prepared a briefing script for volunteers?**

The introductory script reads as follows:

"As part of my dissertation on reducing news overload, I'm conducting a research study into how young people respond to different news presentation formats.

I have a brief survey and consent form which should take about two minutes to complete. The study consists of three questions related to your news habits, then fifteen minutes spent exploring some news content using two different interfaces and answering questions on those.

In total, the study should take around twenty minutes. During the study, your voice will be recorded. Afterwards, the recording will be anonymously named and securely stored. No other personally identifying information is being collected.

Your participation in this study is completely voluntary, you may skip any questions that you don't want to answer, or withdraw at any point.

Thank you for your participation. Do you have any questions?"

Q.2 **Will the participants be using any non-standard hardware?**

The study does not involve the use of any non-standard hardware.

Q.3 **Is there any intentional deception of the participants?**

The study does not involve any deception of participants.

Q.4 **How will participants voluntarily give consent?**

Participants will be asked to sign a consent form (See Appendix C.2) and will be able to withdraw at any time during or after the study.

Q.5 **Will the participants be exposed to any risks greater than those encountered in their normal work life?**

The study does not involve any exposure to risk.

Q.6 **Are you offering any incentive to the participants?**

No incentive will be offered to participants.

Q.7 **Are any of your participants under the age of 16?**

All participants will be over the age of 18 so none will require parental consent.

Q.8 **Do any of your participants have an impairment that will limit their understanding or communication?**

This study requires that all participants have perfect or corrected to perfect vision and hearing. Subjects with other communication impairments will not be eligible to participate in the study.

Q.9 **Are you in a position of authority or influence over any of your participants?**

No, nor have I ever been.

Q.10 **Will the participants be informed that they could withdraw at any time?**

The introductory script informs participants of this.

Q.11 **Will the participants be informed of your contact details?**

All participants will be given my contact details before the study.

Q.12 **Will participants be de-briefed?**

The debriefing script reads as follows: "This study was conducted in order to evaluate people's understanding of news articles when they were presented in a graphically structured manner, versus a traditional list format. I want to test the hypothesis that news consumers feel less overloaded when given an overview of the news in the form of a metro map.

I anticipate that overall, subjects will have identified a broader and longer list of topics in the index task and answered more optimistically in the guessing task using the metro map visualisation as opposed to the list view. I also anticipate that most subjects will use the connections in the map visualisation to help them answer the "most important article" question.

Again, if you wish to contact me in future then you can email me at dtstb20@bath.ac.uk. Thanks for your time."

Q.13 **Will the data collected from the participants be stored in an anonymous form?**

Participants will be recorded using a dictaphone. Each audio recording will be kept with a numerical identifier for each participant, and recordings will be stored digitally with password-protection. No other personally identifying information will be collected.

C.2 User Study Consent Form

Before the study, all participants were asked to read and sign the following consent form, which is based on an example from the University of Nottingham School of Education[1].

PARTICIPANT CONSENT FORM

Newsfeed Visualisation with Metro Maps

Researcher: Damask Talary-Brown (dtstb20@bath.ac.uk)
Supervisor: Professor Stephen Payne

1) The nature and purpose of the study has been explained to me and I agree to participate.

2) I understand the purpose of the research project and my involvement in it.

3) I understand that I may withdraw from the study at any stage and that this will not affect my status now or in the future.

4) I understand that while information gained during the study may be published, I will not be identified and my personal results will remain confidential.

5) I understand that my voice will be recorded during the interview and consent to this.

6) I understand that data will be stored securely and anonymously, where only myself and my supervisor have access to it.

7) I understand that I may contact the researcher or supervisor if I have any further questions or if I wish to make a complaint relating to my involvement in the research.

Participant Name: .....

Signed: ..... Date: .....

C.3 User Study Metro Maps

Both evaluation corpora were extracted from The Guardian's US News RSS Feed in March 2017. The station labelled (a) in each map is the article expanded in the right hand pane.

[1] `http://www.nottingham.ac.uk/educationstudentintranet/researchethics`

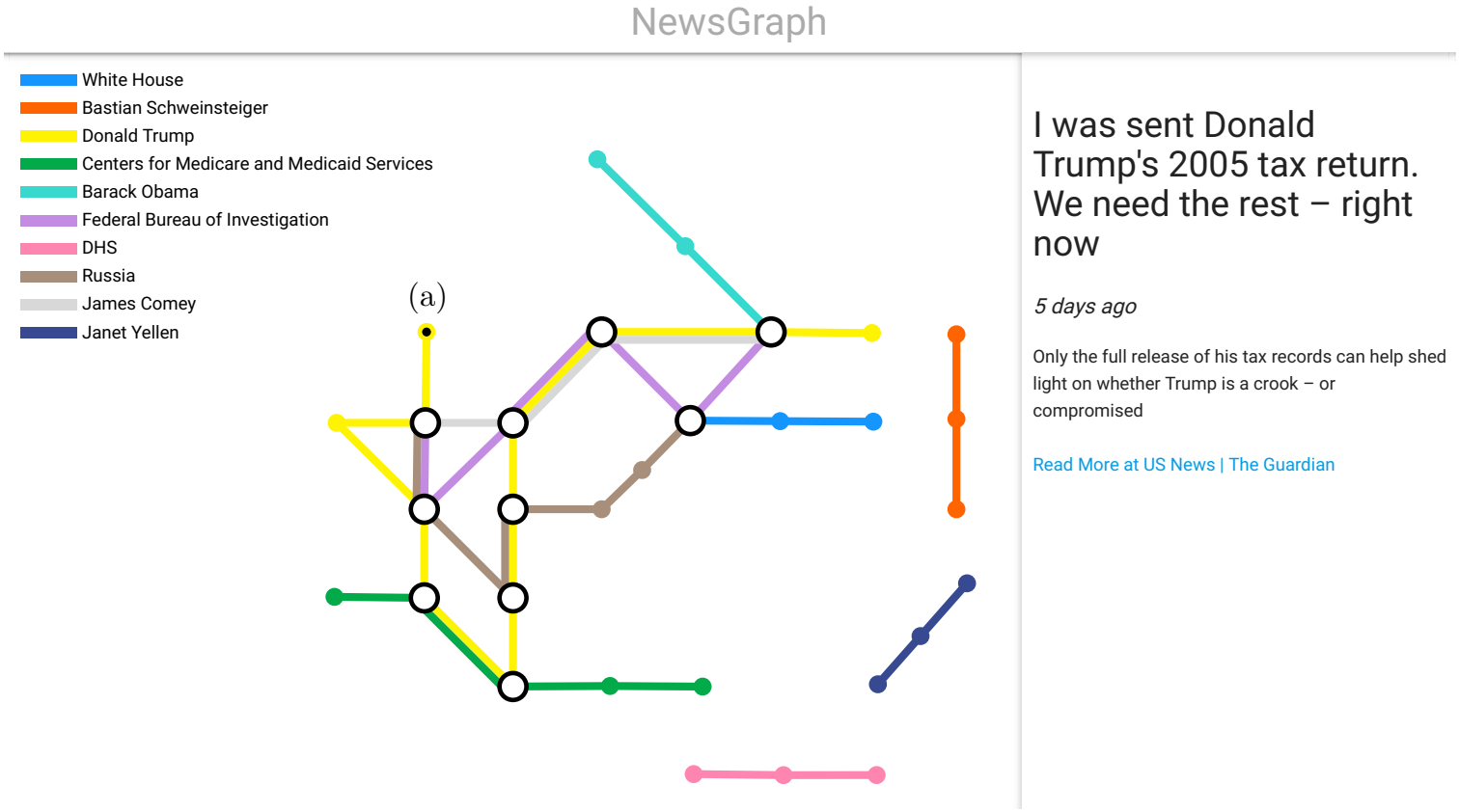


Figure C.1: User Study Data: Map A

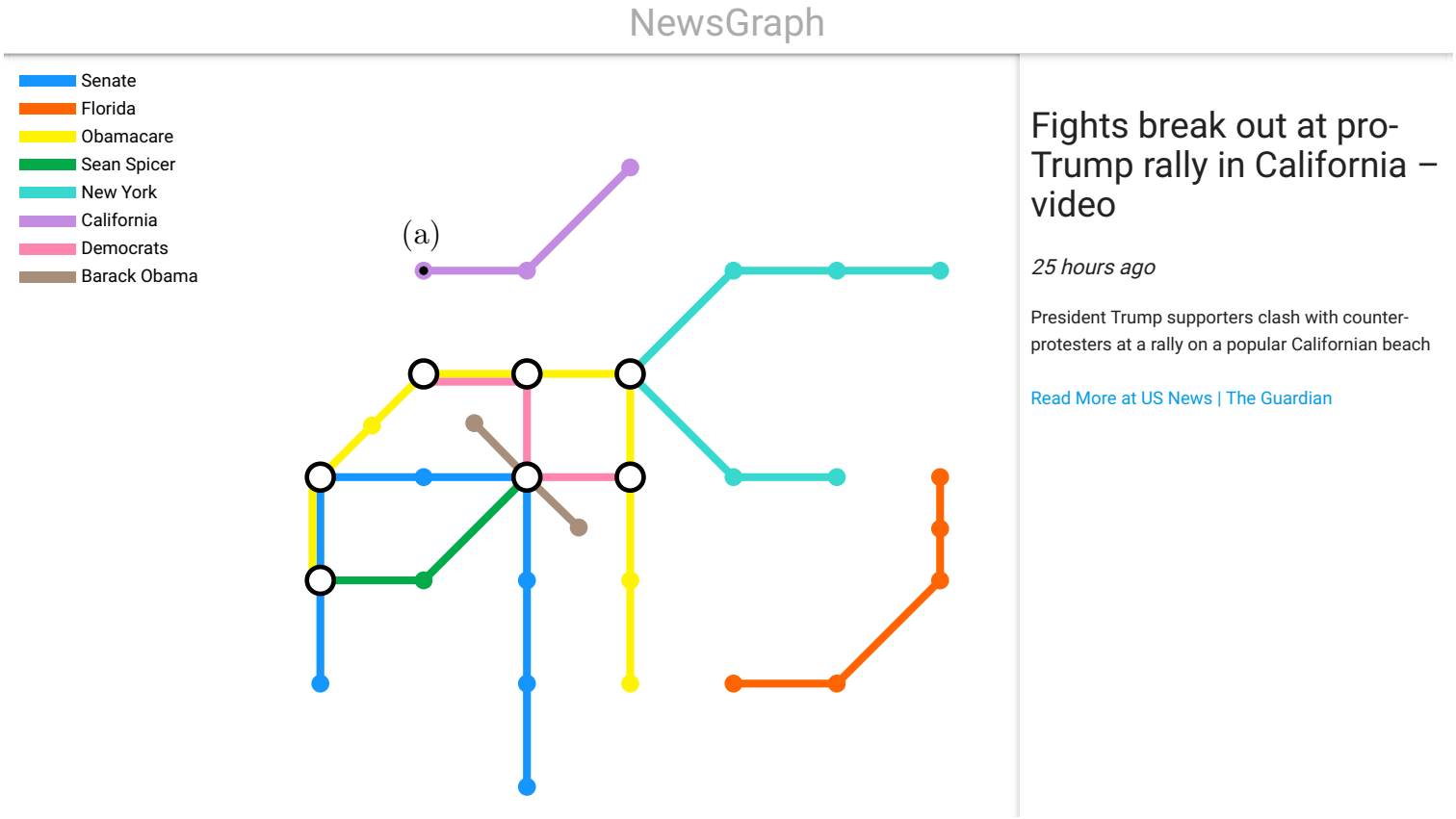


Figure C.2: User Study Data: Map B

C.4 Evaluation Results

Table C.1: Means of *Map Estimate* and *List Estimate*

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Map Estimate	24.75	16	7.188	1.797
	List Estimate	25.88	16	6.820	1.705

Table C.2: Means of *Map Total Topics, Primary Topics, Depth* and *List Total Topics, Primary Topics, Depth*

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Map Total	10.94	16	4.389	1.097
	List Total	8.88	16	2.156	.539
Pair 2	Map Primary	4.19	16	1.601	.400
	List Primary	3.25	16	1.125	.281
Pair 3	Map Depth	2.25	16	.577	.144
	List Depth	2.31	16	.602	.151

Table C.3: Means of *Map Self Assessment Manikin* and *List Self Assessment Manikin*

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Map Pleasure	5.50	16	1.751	.438
	List Pleasure	5.00	16	1.317	.329
Pair 2	Map Arousal	4.19	16	2.287	.572
	List Arousal	4.31	16	1.852	.463
Pair 3	Map Dominance	5.88	16	1.821	.455
	List Dominance	5.63	16	2.217	.554

Table C.4: Results of paired t-test comparing means of Table C.1

	Mean	Std. Deviation	Std. Error Mean	95% CI (Lower)	95% CI (Upper)	t	df	p (2-tailed)
Map Estimate - List Estimate	-1.125	8.936	2.234	-5.887	3.637	-0.504	15	0.622

Table C.5: Results of paired t-test comparing means of Table C.2

	Mean	Std. Deviation	Std. Error Mean	95% CI (Lower)	95% CI (Upper)	t	df	p (2-tailed)
Map Total - List Total	2.063	3.660	0.915	0.112	4.013	2.254	15	0.040
Map Primary - List Primary	0.938	1.879	0.470	-0.064	1.939	1.996	15	0.064
Map Depth - List Depth	-0.063	0.680	0.170	-0.425	0.300	-0.368	15	0.718

Table C.6: Results of paired t-test comparing means of Table C.3

	Mean	Std. Deviation	Std. Error Mean	95% CI (Lower)	95% CI (Upper)	t	df	p (2-tailed)
Map Pleasure - List Pleasure	0.500	2.191	0.548	-0.667	1.667	0.913	15	0.376
Map Arousal - List Arousal	-0.125	1.784	0.446	-1.076	0.826	-0.280	15	0.783
Map Dominance - List Dominance	0.250	2.620	0.655	-1.146	1.646	0.382	15	0.708

Table C.7: Pearson's correlation coefficients between *Weekly News Consumption (Hours)* and *Total/Primary Topics, Depth*

		Avg. Hours	Map Total	Map Primary	Map Depth	List Total	List Primary	List Depth
Avg. Hours	Pearson Correlation	1	.541*	-0.091	.606*	.573*	0.293	0.397
	Sig. (2-tailed)		0.031	0.737	0.013	0.020	0.271	0.127
	N	16	16	16	16	16	16	16
Map Total	Pearson Correlation	.541*	1	0.419	.691**	.556*	0.233	0.159
	Sig. (2-tailed)	0.031		0.106	0.003	0.025	0.386	0.556
	N	16	16	16	16	16	16	16
Map Primary	Pearson Correlation	-0.091	0.419	1	-0.054	-0.186	0.083	-0.272
	Sig. (2-tailed)	0.737	0.106		0.842	0.491	0.759	0.307
	N	16	16	16	16	16	16	16
Map Depth	Pearson Correlation	.606*	.691**	-0.054	1	.616*	0.103	0.336
	Sig. (2-tailed)	0.013	0.003	0.842		0.011	0.705	0.204
	N	16	16	16	16	16	16	16
List Total	Pearson Correlation	.573*	.556*	-0.186	.616*	1	0.069	.546*
	Sig. (2-tailed)	0.020	0.025	0.491	0.011		0.800	0.029
	N	16	16	16	16	16	16	16
List Primary	Pearson Correlation	0.293	0.233	0.083	0.103	0.069	1	-0.320
	Sig. (2-tailed)	0.271	0.386	0.759	0.705	0.800		0.227
	N	16	16	16	16	16	16	16
List Depth	Pearson Correlation	0.397	0.159	-0.272	0.336	.546*	-0.320	1
	Sig. (2-tailed)	0.127	0.556	0.307	0.204	0.029	0.227	
	N	16	16	16	16	16	16	16

*. Correlation is significant at the 0.05 level (2-tailed).

**. Correlation is significant at the 0.01 level (2-tailed).

Table C.8: Pearson's correlation coefficients between *Weekly News Consumption (Hours)* and Self-Assessment Manikin Scores

		Avg. Hours	Map Pleasure	Map Arousal	Map Dominance	List Pleasure	List Arousal	List Dominance
Avg. Hours	Pearson Correlation	1	0.329	0.060	0.364	0.367	-0.031	.508*
	Sig. (2-tailed)		0.214	0.826	0.166	0.162	0.909	0.044
	N	16	16	16	16	16	16	16
Map Pleasure	Pearson Correlation	0.329	1	0.108	.773**	0.000	0.113	0.172
	Sig. (2-tailed)	0.214		0.690	0.000	1.000	0.677	0.525
	N	16	16	16	16	16	16	16
Map Arousal	Pearson Correlation	0.060	0.108	1	0.182	-0.089	.646**	0.159
	Sig. (2-tailed)	0.826	0.690		0.500	0.744	0.007	0.555
	N	16	16	16	16	16	16	16
Map Dominance	Pearson Correlation	0.364	.773**	0.182	1	-0.306	0.269	0.169
	Sig. (2-tailed)	0.166	0.000	0.500		0.249	0.313	0.531
	N	16	16	16	16	16	16	16
List Pleasure	Pearson Correlation	0.367	0.000	-0.089	-0.306	1	-0.328	.685**
	Sig. (2-tailed)	0.162	1.000	0.744	0.249		0.215	0.003
	N	16	16	16	16	16	16	16
List Arousal	Pearson Correlation	-0.031	0.113	.646**	0.269	-0.328	1	-0.164
	Sig. (2-tailed)	0.909	0.677	0.007	0.313	0.215		0.543
	N	16	16	16	16	16	16	16
List Dominance	Pearson Correlation	.508*	0.172	0.159	0.169	.685**	-0.164	1
	Sig. (2-tailed)	0.044	0.525	0.555	0.531	0.003	0.543	
	N	16	16	16	16	16	16	16

*. Correlation is significant at the 0.05 level (2-tailed).

**. Correlation is significant at the 0.01 level (2-tailed).

Table C.9: Pearson's correlation coefficients between *Exuberance, Relaxation* and *Total Topics*

		Map Total	Map Exuberance	Map Relaxation	List Total	List Exuberance	List Relaxation
Map Total	Pearson Correlation	1	.753**	.279	.556*	.469	-.036
	Sig. (2-tailed)		.001	.295	.025	.067	.896
	N	16	16	16	16	16	16
Map Exuberance	Pearson Correlation	.753**	1	.372	.399	.356	-.159
	Sig. (2-tailed)	.001		.156	.126	.176	.556
	N	16	16	16	16	16	16
Map Relaxation	Pearson Correlation	.279	.372	1	.619*	-.111	.099
	Sig. (2-tailed)	.295	.156		.011	.684	.715
	N	16	16	16	16	16	16
List Total	Pearson Correlation	.556*	.399	.619*	1	.258	.274
	Sig. (2-tailed)	.025	.126	.011		.334	.304
	N	16	16	16	16	16	16
List Exuberance	Pearson Correlation	.469	.356	-.111	.258	1	.525*
	Sig. (2-tailed)	.067	.176	.684	.334		.037
	N	16	16	16	16	16	16
List Relaxation	Pearson Correlation	-.036	-.159	.099	.274	.525*	1
	Sig. (2-tailed)	.896	.556	.715	.304	.037	
	N	16	16	16	16	16	16

*. Correlation is significant at the 0.05 level (2-tailed).

**. Correlation is significant at the 0.01 level (2-tailed).

Table C.10: Results of the Shapiro-Wilk test for Normality

Variable	W	df	Sig.
Map Estimate	.881	16	.080
List Estimate	.954	16	.559
Map Total Topics	.958	16	.633
Map Primary Topics	.932	16	.260
Map Index Depth	.746	16	.001
List Total Topics	.917	16	.152
List Primary Topics	.828	16	.007
List Index Depth	.587	16	.000
Map Pleasure (SAM)	.883	16	.043
Map Arousal (SAM)	.925	16	.201
Map Dominance (SAM)	.944	16	.399
List Pleasure (SAM)	.913	16	.131
List Arousal (SAM)	.973	16	.878
List Dominance (SAM)	.916	16	.146

C.5 Raw Participant Data

Table C.11: Participant Background and News Habits

Subject	Group	Gender	Education Level	Weekly News (hours)	RSS Reader
1	1	Female	Undergraduate	10	Currently
2	2	Male	Undergraduate	7	Previously
3	3	Male	Undergraduate	3	Never
4	4	Male	Undergraduate	15	Previously
5	1	Male	Undergraduate	4	Previously
6	2	Female	Undergraduate	1	Never
7	3	Female	Undergraduate	5	Previously
8	4	Male	Undergraduate	2	Previously
9	1	Male	Undergraduate	8	Never
10	2	Male	Undergraduate	40	Previously
11	3	Male	Undergraduate	15	Previously
12	4	Female	Undergraduate	1	Never
13	1	Male	Undergraduate	14	Never
14	2	Male	Undergraduate	10	Never
15	3	Male	Undergraduate	10	Never
16	4	Male	Undergraduate	0	Never

Table C.12: Task Performance Data: *Map*

Subject	Estimate	Index Task			Self-Assessment Manikin		
		Primary	Total	Depth	Pleasure	Arousal	Dominance
1	20	2	11	3	7	2	6
2	40	3	9	2	6	5	5
3	25	4	10	2	7	7	6
4	17	5	12	2	6	7	8
5	18	5	8	2	3	3	6
6	23	3	9	2	2	5	3
7	28	4	8	2	6	4	6
8	30	4	4	1	3	3	4
9	40	8	19	2	7	8	8
10	25	3	14	3	6	3	6
11	20	5	17	3	8	6	9
12	15	2	7	2	6	1	5
13	25	4	16	3	4	7	4
14	20	3	10	2	6	1	7
15	25	6	16	3	7	2	8
16	25	5	5	2	4	3	3
Average	24.75000	4.12500	10.93750	2.25000	5.50000	4.18750	5.87500

Table C.13: Task Performance Data: *List*

Subject	Estimate	Index Task			Self-Assessment Manikin		
		Primary	Total	Depth	Pleasure	Arousal	Dominance
1	25	2	11	3	4	2	4
2	25	2	6	2	5	5	4
3	16	4	9	2	6	3	8
4	20	4	10	2	4	5	7
5	20	4	6	2	3	4	3
6	30	5	8	2	5	6	5
7	23	2	10	2	5	6	4
8	30	2	6	2	4	3	5
9	30	4	9	2	3	7	2
10	30	5	12	3	7	3	8
11	30	4	9	2	5	8	8
12	20	4	10	2	5	4	3
13	40	3	9	2	7	5	9
14	35	2	10	4	7	1	8
15	25	2	12	3	4	4	7
16	15	3	5	2	6	3	5
Average	25.87500	3.25000	8.87500	2.31250	5.00000	4.31250	5.62500

Appendix D
Code

D.1 Project Structure

The structure of the project is as follows, with source files available in the following sections.

```
src/
├── D.2:  newsgraph.py
├── newsUtils/ ..... Classes encapsulating feeds, articles and corpora
│   └── D.3:  newsfeeds.py
├── keywordUtils/ ..... Keyword extraction and entity disambiguation
│   ├── D.4:  corpusUtilities.py
│   └── D.5:  graphKnowledge.py
├── ioUtils/ ..... Intermediate cache seralisation
│   └── D.6:  newsSerialiser.py
├── mapEngine/ ..... Boundary between Build Graph and Draw Map
│   └── D.7:  graphObjects.py
└── webUtils/ ..... HTML generation and core JavaScript
    ├── D.8:  html.py
    ├── D.9:  newsgraph.js
    └── D.10:  newsgraph.css
```

1 Installation

To install the project's dependencies, navigate to the `src` directory and optionally, activate a `virtualenv`. Install with:

```
$ pip install -r requirements.txt
```

2 Running

Specify one or more newsfeeds in `newsgraph.py:36`. Optionally, specify the lists *interests* and *ignore* on the lines below. Run NewsGraph with:

```
$ python newsgraph.py
```

When the script terminates, the file `ng.html` in the `src` directory will contain the metro map and all of its associated data.

D.2 File: src/newsgraph.py

```
import datetime
import feedparser
import newspaper
import operator

from itertools import combinations
from time import strftime

from newsUtils.newsfeeds import *
from ioUtils.newsSerialiser import *
from mapEngine.graphObjects import *

class Params:
  # Too many stations? Reduce this.
  RSS_LIMIT = 50

  # Stations too highly connected? Reduce this. (Vice versa for too sparse)
  TFIDF_VEC_LEN = 7
  # Metro lines too vague? Increase this.
  CORPUS_KEYWORDS = 25

  #Too many metro lines? Reduce this.
  METRO_LINES = 10 #
  # Too few metro lines? Reduce this.
  MIN_LINE_LENGTH = 4
  # Map empty due to too few common keywords? Invert this.
  TFIDF = True
  TFPDF = not TFIDF

read_cache = True
read_cache_name = None # eg. old-cache.ng in the cwd

write_cache = False
write_cache_name = None # eg. my-cache.ng

feeds = [] # eg. ['https://www.theguardian.com/us-news/rss', ...]
interests = []
ignore = []

cln = None
if read_cache:
  rd = CacheReader(read_cache_name)
  cln = rd.read()
else:
  urls = []
  print "Fetching RSS Feed(s)"
  for feed in feeds:
    feed = feedparser.parse(feed)
    for item in feed["items"][:Params.RSS_LIMIT]:
      urls.append(
        (item["link"],
        item["published_parsed"],
        feed["channel"]["title"],
        item.get("author", "")))
  cln = NewsCollection(urls)

if write_cache:
  wr = CacheWriter(write_cache_name)
  wr.write(cln)

top_keywords = None
if Params.TFIDF:
  e_map = cln.top_article_keywords(
    Params.TFIDF_VEC_LEN, include=interests, exclude=ignore)
  top_keywords = sorted(
    e_map.iteritems(), key=lambda (k,a): len(a), reverse=True
  )[:Params.CORPUS_KEYWORDS]
else: #tf-pdf
  top_keywords = sorted(
    cln.top_corpus_keywords(
      Params.CORPUS_KEYWORDS, include=interests, exclude=ignore
    ).iteritems(), key=lambda (k,a): len(a), reverse=True)

# Compute coverage for each keyword in the corpus
coverage = {}
for keyword, articles in top_keywords:
  for a in articles:
    if a in coverage: # This article is already on one or more lines
      coverage[a].append(keyword)
    else: # It's the first time we've seen this article
      coverage[a] = [keyword]
```

```

# Now coverage[article] will tell us which lines it is on,
# we penalise lines which cover already-covered articles.
value_added = dict((keyword, 0) for keyword,_ in top_keywords)
for keyword, articles in top_keywords:
  for a in articles:
    weight = len(coverage[a])
    value_added[keyword] += cln.tf_idf(keyword, a, id=False)/weight
  value_added[keyword] *= len(articles)

# Compute the affinity for each keyword
affinity = dict((keyword, 0) for keyword,_ in top_keywords)
for keyword, articles in top_keywords:
  line_affinity = dict((o, 0) for o,_ in top_keywords)
  del line_affinity[keyword] #Can't compute a line's affinity with itself
  for a in articles:
    others = [c for c in coverage[a] if c != keyword]
    for o in others:
      line_affinity[o] += 1
  for o, val in line_affinity.iteritems():
    line_affinity[o] = (2**val)
  affinity[keyword] = sum(line_affinity.values())/len(articles)

top_keywords = sorted(
  top_keywords,
  key=lambda (e,a): value_added[e]/(max(1, affinity[e])),
  reverse=True
)[:Params.METRO_LINES]

# Nodes and Links will be the two JSON objects passed to d3.
links = []
nodes = []

metro_lines = {}
node_lookup = {}
multiedge_counts = {}
stations = set()
termini = set()

for i, (e, articles) in enumerate(top_keywords):
  if len(articles) >= Params.MIN_LINE_LENGTH:
    ordered_line = sorted(articles, key=lambda a:a.publish_date)
    # Add the articles to node_lookup if they aren't already - this is a
    # simple 0 to (n-1) index of all n articles which will be in the graph.
    # Also, update the lines passing through station a to include this one.
    for n, a in enumerate(ordered_line):
      node_lookup[a] = node_lookup.get(a, len(node_lookup.keys()))
      a.metro_lines.append(e)
      stations.add(a)

    metro_lines[e] = []
    termini.add(node_lookup[ordered_line[0]])
    termini.add(node_lookup[ordered_line[-1]])
    for n in range(len(ordered_line)-1):
      source = node_lookup[ordered_line[n]]
      target = node_lookup[ordered_line[n+1]]
      count = multiedge_counts.get((source, target), 0)+1
      multiedge_counts[(source, target)] = count

      links.append({
        "source": source,
        "target": target,
        "count": count,
        "line": e,
      })
      metro_lines[e].append((source, target))

for a,i in sorted(node_lookup.iteritems(), key=lambda x:x[1]):
  nodes.append({"name": a.title, "lines":a.metro_lines, "data":a.data()})
for i, node in enumerate(nodes):
  node["terminus"] = (i in termini)

# Generate the graph
ng = NewsGraph(nodes, links, stations, metro_lines, node_lookup)
```

D.3 File: src/newsUtils/newsfeeds.py

```
import datetime
import math
import newspaper
import operator
import pytz
import string
import uuid

from collections import Counter
from newspaper import Article as RawArticle
```

```
from time import mktime
from tqdm import tqdm

from keywordUtils.corpusUtilities import *

class NewsFeed:
  """
  Container class for instances of Article; used to calculate tf-pdf metrics.
  """
  def __init__(self, feed_name):
    """
    Initialise NewsFeed from a single RSS feed.
    """
    self.name = feed_name
    self.F_jc = Counter()  # Frequency of term within articles in this feed
    self.n_jc = Counter()  # Articles in the feed which contain this term
    self.article_count = 0 # Total articles in the feed

  def addArticle(self, article):
    """
    Add a new article to the feed and update the tf-pdf metrics.
    """
    self.finalised = False
    self.article_count += 1
    for e in article.entities:
      self.n_jc[e] += 1
      self.F_jc[e] += article.termFrequency(e)

  def finalise(self):
    """
    Finalise the feed for tf-pdf calculations.
    This must be called again if any new articles are added.
    """
    self.finalised = True
    self.norm_tf = sum(self.F_jc[k]**2 for k in self.n_jc.elements())
    self.N_c = float(self.article_count)

  def term_weighting(self, term):
    """Calculate the term weighting metric for tf-pdf in this channel."""
    if not self.finalised:
      self.finalise()
    norm_F_jc = self.F_jc[term]/math.sqrt(self.norm_tf)
    pdf = math.pow(math.e, (self.n_jc[term]/self.N_c))
    return norm_F_jc * pdf


# Don't sublclass from newspaper.article as objects instantiated from
# this class must be serialisable to the cache using cPickle (i.e. no lxml)
class Article:
  """
  Serialisable wrapper around Newspaper.Article, with meta from Goose.
  """
  def __init__(self, article, id, publish_date, author, feed):
    """
    Create a new instance of Article from params. This will perform
    text & date parsing, tokenisation and entity disambiguation.
    """
    self.id = id
    self.title = article.title
    self.author = author
    self.html = article.article_html
    self.text, self.summary = parseHTML(article)
    self.img = article.top_img
    self.url = article.url
    self.feed_name = feed
    self.metro_lines = []
    self.line_idx = {}
    self.tokens, self.entities, self._ent_frequency = tokenize(
      ".".join(self.title, self.text))
    self.entities = list(
      set(self.entities).difference(self.author, *self.feed_name.split()))
    self.word_count = len(self.tokens)
    self.term_frequency = {}
    try:
      self.publish_date = publish_date.replace(tzinfo=pytz.UTC)
    except AttributeError as e:
      self.publish_date = publish_date

  def termFrequency(self, term):
    """
    Calculate the tf of {term} in this article
    """
    if self._ent_frequency[term]:
      return self._ent_frequency[term]

    if not self.term_frequency:
      for word in self.tokens + self.entities:
        current = self.term_frequency.get(word, 0)
        self.term_frequency[word] = current+1

    freq = self.term_frequency.get(term, 0)
```

```
    if freq == 0:
      return self.text.lower().count(term.lower())
    else:
      return freq

  def containsTerm(self, term):
    """
    Boolean check of {term} in article
    """
    return term.lower() in self.text.lower() or self._ent_frequency[term]

  def data(self):
    """
    Return a JSON object containing the pane data for this article/station
    """
    html = (self.summary + "<br/><br/><a href='%s'>Read more at %s</a>" %
      (self.url, self.feed_name))
    return {
      "title": self.title,
      "date": 1000*mktime(self.publish_date),
      "summary": self.summary,
      "img": self.img,
      "url": self.url,
      "html": html
    }

class NewsCollection:
  """
  Container class for instances of NewsFeed. Used for corpus-wide metrics.
  """
  def __init__(self, feed_data):
    """
    Create a new instance of NewsCollection-(1:m)-NewsFeed-(1:m)-Article
    """
    self.collection_by_id = {}
    self.collection_by_feed = {}
    self.article_count = len(feed_data)

    self.newsfeeds = {}
    for feed_name in set(f[2] for f in feed_data):
      feed = NewsFeed(feed_name)
      self.newsfeeds[feed_name] = feed
      self.collection_by_feed[feed] = []

    for (url, publish_date, feed_name, author) in tqdm(feed_data):
      article = RawArticle(url, keep_article_html=True)
      article.download()
      article.parse()
      article_id = uuid.uuid1()
      processed_article = Article(
        article, article_id,
        publish_date, author, feed_name)
      self.collection_by_id[article_id] = processed_article
      self.newsfeeds[feed_name].addArticle(processed_article)

    self.feed_count = len(self.collection_by_feed)

  def article(self, article_id):
    """
    Get the article object by its id.
    """
    return self.collection_by_id.get(article_id, None)

  def idf(self, term):
    """
    Calculate the inverse document frequency for {term} in this corpus.
    """
    n_containing = 0
    for article in self.collection_by_id.values():
      if article.containsTerm(term):
        n_containing += 1
    return math.log(self.article_count/n_containing)

  def tf_idf(self, term, article, id=True):
    """
    Calculate the tf-idf for {term} in {article} in this corpus.
    {article} can either be by val or by id.
    """
    if id:
      article = self.article(article)
    try:
      return article.termFrequency(term) * self.idf(term)
    except ZeroDivisionError:
      pass # Probably a whitespace/punctuation error
      #raise KeyError('Keyword "%s" does not exist in any article' % term)

  def entity_tf_idf_vector(self, article_id, top_n=-1, include=[], boost=10):
    """
    Return the top n entities in this article by descending tf-idf.
    Default n = all.
```

```
    Boost the score of any terms in {include} by {boost, default=10}
    """
    vector = {}
    article = self.article(article_id)
    for i, term in enumerate(article.entities):
      tfidf = self.tf_idf(term, article_id)
      if tfidf:
        vector[term] = tfidf
        if term in include:
          vector[term] *= boost
    return dict(sorted(
        vector.iteritems(), key=operator.itemgetter(1), reverse=True
      )[:top_n])

  def tf_pdf(self, term):
    """
    Calculate the tf-pdf of {term} in this corpus
    """
    return sum(feed.term_weighting(term) for feed in self.newsfeeds.values())

  def top_article_keywords(self, top_n=10, include=[], exclude=[]):
    """
    Return a dictionary of {entities:containing articles} in this Article
    """
    ret = {}
    for id, article in self.collection_by_id.iteritems():
      for entity in self.entity_tf_idf_vector(id, top_n, include):
        if entity not in exclude:
          update = ret.get(entity, [])
          update.append(article)
          ret[entity] = update
    return ret

  def top_corpus_keywords(self, top_n=25, include=[], exclude=[], boost=10):
    """
    Return a dictionary of the top {entity:[(score,article),...]} in this
    corpus, with words in {include} boosted and words in {exclude} ignored.
    """
    ret = {}
    corpus_entities = set(
      e for a in self.collection_by_id.values() for e in a.entities
      if e not in exclude)

    for entity in corpus_entities:
      update = ret.get(entity, [])
      tfpdf = self.tf_pdf(entity)
      if entity in include:
        tfpdf *= boost
      ret[entity] = (tfpdf,
        [a for a in self.collection_by_id.values()
        if entity in a.entities])

    return dict((k, v[1]) for (k, v) in sorted(
      ret.iteritems(), key=operator.itemgetter(1), reverse=True)[:top_n])
```

D.4 File: src/keywordUtils/corpusUtilities.py

```
import nltk
import string
import unicodedata

from lxml import etree

from cStringIO import StringIO
from stop_words import get_stop_words
from sys import maxunicode
from unidecode import unidecode
from goose import Goose

from graphKnowledge import NewsGraphKnowledge

gooseExtractor = Goose()

def tokenize(doc, knowledge=True, ltze=False):
  """
  Tokenise the article, extract and disambiguate its named entities.
  """
  wnl = nltk.WordNetLemmatizer()
  knowledge = NewsGraphKnowledge()
  tokens = []
  entities = []
  # Don't transform letter case until after entity recognition
  punctuation = [i for i in xrange(maxunicode)
    if unicodedata.category(unichr(i)).startswith('P')]

  stopWords = set(get_stop_words("en"))
  prevTree = ""
```

```
  for sentence in preprocess(doc):
    if prevTree:
      tokens.append(prevTree.strip())
      entities.append(prevTree.strip())
      prevTree = ""
    for chunk in nltk.ne_chunk(sentence, binary=False):
      if isinstance(chunk, nltk.Tree):
        t = " ".join(ent[0] for ent in chunk.leaves()).strip()
        prevTree += t+" "
      else:
        # Keep accumalating named entity chunks and appending, until
        # we reach a non-NE chunk which signals the end of the previous.
        if prevTree:
          tokens.append(prevTree.strip())
          entities.append(prevTree.strip())
          prevTree = ""
        t = chunk[0].lower()
        if ltze:
          t = wnl.lemmatize(t)
        if t not in stopWords and not t.isdigit():
          t = string.strip(t, string.punctuation)
          tokens.append(t)

  aliases, freqs = knowledge.aliasEntities(entities)
  entities = [aliases.get(e, e) for e in entities]
  return tokens, entities, freqs

def parseHTML(article):
  """
  Get cleaned article text and meta discription from url
  """
  article = gooseExtractor.extract(url=article.url)
  return article.cleaned_text, article.meta_description

def preprocess(doc):
  """
  Sentence tokenise then word tokenise an article
  """
  sentences = nltk.sent_tokenize(doc)
  words = [nltk.word_tokenize(s) for s in sentences]
  return [nltk.pos_tag(w) for w in words]
```

D.5 File: src/keywordUtils/graphKnowledge.py

```
import json
import urllib

from collections import Counter
from itertools import combinations
from sys import maxint

class NewsGraphResult:
  """
  Class encapsulating result XML from Google's knowledgegraph API
  """
  NAME  = 0
  TYPES = 1
  DESCR = 2
  SCORE = 3

  def __init__(self, query, candidates):
    """
    Extract the fields we want - name, description, and score.
    """
    self.query = query
    self.candidates = []
    for response in candidates:
      desc = response['result'].get('description', "")
      if 'detailedDescription' in response['result']:
        desc = response['result']['detailedDescription']
        desc = desc[u'articleBody'].encode('utf-8')
      self.candidates.append((
        response['result'].get('name', query),
        response['result']['@type'],
        desc,
        response['resultScore'],
      ))
    if len(candidates) < 2:
      self.certainty = 0
    else:
      first = candidates[0]['resultScore']
      second = candidates[1]['resultScore']
      self.certainty = first/second

  def nth(self, n):
    """
```

```
    Get the nth result for this query
    """
    if self.candidates and len(self.candidates) > n-1:
      return self.candidates[n-1]
    else:
      return None

  def top(self):
    return self.nth(1)

  def isPerson(self):
    """
    Make an educated guess as to whether this entity is a person
    """
    if self.candidates:
      return "Person" in self.candidates[0][NewsGraphResult.TYPES]
    return None


class NewsGraphKnowledge:
  """
  Class for performing entity disambiguation with knowledge graph
  """
  LOG = False
  IGNORE = [u"AP", u"Caption", u"Getty Images Image", u"Getty Images",
      u"Getty", u"Photograph", u"Photo", u"Facebook Twitter Pinterest"]
  API_KEY = 'keywordUtils/.api_key'

  def __init__(self):
    """
    Load the API key so we can make requests
    """
    self.api_key = open(NewsGraphKnowledge.API_KEY).read()
    self.service_url = 'https://kgsearch.googleapis.com/v1/entities:search'

  def query(self, queryText, limit=3):
    """
    Return the results of a query against the API
    """
    params = {
      'query': queryText,
      'limit': limit,
      'indent': True,
      'key': self.api_key,
    }
    url = self.service_url + '?' + urllib.urlencode(params)
    response = json.loads(urllib.urlopen(url).read())

    if 'itemListElement' in response:
      return NewsGraphResult(queryText, response['itemListElement'])
    else:
      return None

def popular(popular_entities, result):
  return result.top()[NewsGraphResult.NAME] in popular_entities

def aliasEntities(self, entities, certainty=3):
  """
  Generate the set of unambiguous aliases for entities in an article
  """
  uncertain = []
  mapping = {}
  people = {}

  strength = {e: entities.count(e) for e in entities}

  popular_entities = list(set(
    [e for e in entities
    if entities.count(e) > 2 and len(e)
    and e not in NewsGraphKnowledge.IGNORE]
  ))

  out = set()
  for e1, e2 in combinations(popular_entities, 2):
    if len(e1) > len(e2):
      tmp = e1
      e1 = e2
      e2 = tmp
    if e1 in e2:
      if e1 in mapping:
        del mapping[e1]
        out.remove(e1)
      else:
        mapping[e1] = e2
        out.add(e1)

  popular_entities = [e for e in popular_entities if e not in out]

  entity_set = list(set(entities))
  entity_set.sort(key=lambda e:len(e), reverse=True)
```

```
133
134     for queryText in entity_set:
135       if isinstance(queryText, unicode):
136         queryText = queryText.encode('utf-8')
137       elif isinstance(queryText, str):
138         queryText.decode('utf-8')
139       result = self.query(queryText)
140
141       if result==None or result.top()==None:
142         continue
143
144       if result.certainty > certainty or popular(popular_entities, result):
145         if NewsGraphKnowledge.LOG:
146           n = result.top()[NewsGraphResult.NAME]
147           if isinstance(n, unicode):
148             n = n.encode('utf-8')
149           elif isinstance(n, str):
150             n.decode('utf-8')
151           print "Certain: %s => %s" % (queryText, n)
152
153         name = result.top()[NewsGraphResult.NAME]
154         if name in people:
155           mapping[queryText] = people[name]
156         else:
157           mapping[queryText] = name
158           if result.isPerson() and " " in name:
159             for word in name.split(" "):
160               if word in people and people[word] != name:
161                 if NewsGraphKnowledge.LOG:
162                   print "Ambigiuty, ignoring", name
163                 else:
164                   people[word] = name
165       else:
166         if queryText in people:
167           mapping[queryText] = people[queryText]
168         else:
169           mapping[queryText] = queryText
170
171     aliases = dict(
172       (entity, alias) for entity, alias in mapping.iteritems()
173       if entity!=alias
174     )
175     freqs = Counter([aliases.get(e, e) for e in entities])
176     return aliases, freqs
```

D.6 File: src/ioUtils/newsSerialiser.py

```
1  try:
2    import cPickle as pickle
3  except:
4    import pickle
5
6  class Cache:
7    DEFAULT = "/io_ignore/cache.ng"
8    def __init__(self, file_name=DEFAULT):
9      self.fname = file_name
10
11 class CacheWriter(Cache):
12   def write(self, cln):
13     with open(self.fname, 'wb') as cache:
14       pickle.dump(cln, cache)
15
16 class CacheReader(Cache):
17   def read(self):
18     cln = None
19     with open(self.fname, 'rb') as cache:
20       cln = pickle.load(cache)
21     return cln
```

D.7 File: src/mapEngine/graphObjects.py

```
1 from webUtils.html import writePage
2
3 class NewsGraph:
4   def __init__(self, nodes, links, articles, lines, indexes):
5     writePage(nodes, links, articles, lines)
```

D.8 File: src/webUtils/html.py

```
1 # -*- coding: latin-1 -*-
2 from json import dumps
3 from lxml import etree
```

```
from lxml.builder import E
from time import mktime, strftime
from datetime import datetime
from xml.sax.saxutils import escape

class Libraries:
  MATERIALIZE = "https://cdnjs.cloudflare.com/ajax/libs/"+ \
  "materialize/0.98.0/css/materialize.min.css"
  MATERIAL_ICONS = \
  "https://fonts.googleapis.com/icon?family=Material+Icons"

  D3 = "http://d3js.org/d3.v2.min.js?2.9.6"
  JQUERY = "https://code.jquery.com/jquery-3.1.1.min.js"
  JQUERY_UI = "http://code.jquery.com/ui/1.11.0/jquery-ui.min.js"

def CLASS(*args):
  return {"class":' '.join(args)}

def buildCards(articles):
  """
  Build the "feed" pane of the web view
  """
  cards = []
  for article in articles:
    card = E.div(
      CLASS("card horizontal",
        *[l.replace(" ","-") for l in article.metro_lines]
      ),
      E.div(
        CLASS("card-image"),
        E.img(src=article.img)
      ),
      E.div(
        CLASS("card-stacked"),
        E.div(
          CLASS("card-content"),
          E.h4(article.title),
          E.p(article.summary)
        ),
        E.div(
          CLASS("card-action"),
          strftime("%c", article.publish_date),
          " - ",
          E.a(
            "Read more at "+ article.feed_name,
            href="/#"#article.url
          )
        )
      )
    )
    cards.append(card)
  return cards


def writePage(nodes, links, articles, metro_lines):
  """
  Generate the page, insert our station data JSON at the placeholders.
  """
  with open("webUtils/newsgraph.css", "r") as css:
    default_css = css.read()
  with open("webUtils/newsgraph.js", "r") as js:
    default_js = js.read()
    default_js = default_js.replace("NG-NODES", dumps(nodes))
    default_js = default_js.replace("NG-LINKS", dumps(links))
    default_js = default_js.replace("NG-METRO-LINES", dumps(metro_lines))

  page = (
    E.html(
      E.head(
        E.title("NewsGraph"),
        E.link(
          href=Libraries.MATERIALIZE,
          rel="stylesheet",
          type="text/css"
        ),
        E.link(
          href=Libraries.MATERIAL_ICONS,
          rel="stylesheet",
          type="text/css"
        ),
        E("script", "", src=Libraries.D3),
        E("script", "", src=Libraries.JQUERY),
        E("script", "", src=Libraries.JQUERY_UI),
        E.style(default_css)
      ),
      E.body(
        E("nav",
          E.div(
            CLASS("nav-wrapper yellow darken-3"),
            E.span(CLASS("brand-logo center"), "NewsGraph"),
```

```
            E.ul(
              CLASS("right hide-on-med-and-down"),
              E.li(E.a("Graph", href="#"),
                CLASS("active"), id="graphPane"
              ),
              E.li(E.a("Feed", href="#"), id="feedPane"),
              id="nav-mobile"
            )
          )
        ),
        E.div(
          *buildCards(articles),
          id="cards"
        ),
        E("script", "NG-JS")
      )
    )
  )

  with open("ng.html", "w") as ng:
    ng.write(etree.tostring(
      page,
      pretty_print=True,
      encoding="ISO-8859-1",
      doctype="<!DOCTYPE html>").replace("NG-JS", default_js
    ))
```

D.9 File: src/webUtils/newsgraph.js

```
var width = window.innerWidth * 0.7,
    height = window.innerHeight-60;

// How energetically is the map moving into position?
var energy;
// How many stations couldn't be positioned?
var unplaced = 0;
// Position hash
var taken = {};
// Size of one grid square
var spacing = height/8.0;
// Has a metro line on the key been clicked?
var iso_toggle = 0;

// Realistically more colours than we will ever need
var color = color = d3.scale.ordinal()
  .domain(50)
  .range(["#1596ff", "#ff6401", "#fef302", "#03aa4a", "#38d8ce",
          "#c38ce2", "#fd85b0", "#a78f7b", "#d8d8d8", "#374a91",
          "#48e268", "#ffa103", "#ff576d", "#66c1fe", "#97f1a0",
          "#3fc104", "#febbc0", "#8494ad"]);
var radius = d3.scale.sqrt().range([0, 6]);

var svg = d3.select("body")
    .append("svg")
    .attr("id", "graph")
    .attr("width", width)
    .attr("height", height)
    .attr("viewBox", "0 0 " + width + " " + height )
    .attr("preserveAspectRatio", "xMidYMid meet")
    .attr("pointer-events", "all");

var article_container = d3.select("body")
    .append("div")
    .attr("id", "article-container")
    .attr("width", window.innerWidth * 0.3)
    .attr("height", height);

var force = d3.layout.force()
    .size([width, height-60])
    .charge(function(d){
        // Nodes with higher weights will repel more strongly
        return d.weight == 1? -300: -500;
    })
    .linkDistance(function(d){
        return height/15;
    })
    .linkStrength(1)
    .friction(0.96)
    .gravity(0.15)
    .alpha(1);

// NG things get regex replaced with actual JSON by the python script
var metro_lines = NG-METRO-LINES;
var graph = {
    "nodes": NG-NODES ,
    "links": NG-LINKS
};
```

```
59
60 // Dates get serialised as plain ints
61 for (var n=0; n<graph.nodes.length;++n) {
62   graph.nodes[n].data.date = new Date(graph.nodes[n].data.date);
63 }
64
65 // Keep a record of which links have multiple metro lines
66 var multiEdges = {};
67 for (var l of graph.links) {
68   multiEdges[[graph.nodes[l.source], graph.nodes[l.target]]] =
69     Math.max(l.count,
70              multiEdges[[graph.nodes[l.source],
71              graph.nodes[l.target]]]||1);
72 }
73
74 // Helper functions for station movement, calculation and manipulation
75 function isInterchange(d) { return d.lines.length > 1; }
76 function linelength(p1, p2){
77   return Math.sqrt(Math.pow(p1.x-p2.x,2) + Math.pow(p1.y-p2.y,2));
78 }
79 function nan(v) { return v !== v; }
80 function coordinates(p) {
81   return JSON.stringify({x:Math.round(p.x, 2),y:Math.round(p.y, 2)});
82 }
83 function octilinear(p1, p2) {
84   if (p1.y == p2.y) {
85     return true;
86   } else if (p1.x == p2.x) {
87     return true;
88   } else {
89     return Math.abs((p2.x-p1.x)/(p2.y-p1.y)) < 1;
90   }
91 }
92 function gradient(a, b){ return (b.py-a.py)/(b.px-a.px); }
93 function dist2d(a, b){ return {x: (b.px-a.px), y:(b.py-a.py)}; }
94
95 // All the d3 drawing stuff and main loop
96 var drawGraph = function (graph) {
97
98   force.nodes(graph.nodes)
99     .links(graph.links)
100    .on("tick", function() {
101        // Update the energy map's energy according to the aesthetic
102        // criteria. Poor aesthetics => More energy to reposition.
103        energy = Math.log(octilinearity())/10 +
104                 Math.log(lineStraightness())/10;
105      force.alpha(energy);
106      tick();
107  })
108  .start();
109
110 // Create the metro line links
111 var link = svg.selectAll(".link")
112   .data(graph.links)
113   .enter().append("path")
114   .style("stroke", function(d) {
115     return color(d.line);
116   })
117   .style("stroke-width", function(d) {
118     return 7;
119   })
120   .attr("class", function (d) {
121     return "link count" + d.count;
122   });
123
124 // Bind the stations to their data
125 var node = svg.selectAll(".node")
126   .data(graph.nodes)
127   .enter().append("g")
128   .attr("class", "node")
129   .on("click", function(d){
130     // Reset all the circles
131     var nodes = d3.selectAll(".node");
132     nodes.each(function(d) {
133       d3.select(this).select("circle").transition().duration(250)
134         .style("fill", isInterchange(d)? "white" :color(d.lines[0]))
135       });
136       d3.select(this).select("circle").transition().duration(250)
137       .style("fill", "black")
138       // __.data__.data is not a d3 thing
139       // (the last .data element is ours and poorly named)
140       showArticle(this.__data__.data);
141     })
142   .call(force.drag);
143
144 // Create the station tooltips
145 node.append("title").text(function(d) { return d.name; });
146
147 // Add the station circles and conditionally colour
148 node.append("circle")
```

```
149   .attr("r", function (d) {
150     return isInterchange(d)? radius(4): radius(1);
151   })
152   .style("stroke", function (d) {
153       return isInterchange(d) ? "black" : color(d.lines[0]);
154   })
155   .style("fill", function (d) {
156       return isInterchange(d) ? "white" : color(d.lines[0]);
157   });
158
159   // Animate
160   function tick() {
161
162     //http://webiks.com/d3-js-force-layout-straight-parallel-links
163     function multiTranslate(targetDistance, point0, point1) {
164       var x1_x0 = point1.x - point0.x,
165           y1_y0 = point1.y - point0.y,
166           x2_x0, y2_y0;
167       if (y1_y0 === 0) {
168           x2_x0 = 0;
169           y2_y0 = targetDistance;
170       } else {
171           var angle = Math.atan((x1_x0) / (y1_y0));
172           x2_x0 = -targetDistance * Math.cos(angle);
173           y2_y0 = targetDistance * Math.sin(angle);
174       }
175       return {dx: x2_x0, dy: y2_y0};
176     }
177
178     // Change the angle and position of the links as they move
179     link.attr("d", function (d) {
180       var x1 = d.source.x, y1 = d.source.y,
181           x2 = d.target.x, y2 = d.target.y,
182           dr = 0;
183           var lw = 7;
184           var offset = multiTranslate(
185             d.count==1?0:(Math.floor(d.count/2))*(d.count%2?-lw:lw),
186             d.source, d.target
187           );
188           x1 += offset.dx;
189           x2 += offset.dx;
190           y1 += offset.dy;
191           y2 += offset.dy;
192       return "M"+x1+","+y1+"A"+dr+","+dr+" 0 0,1"+x2+","+y2;
193     });
194
195     // Change node positions as they move, don't let them go off canvas
196     node.attr("transform", function (d) {
197       return "translate(" + d.x + "," + d.y + ")";
198     });
199     node.attr("cx", function(d) {
200         return d.x = Math.max(10, Math.min(width-10, d.x));
201     })
202     .attr("cy", function(d) {
203       return d.y = Math.max(10, Math.min(height-10, d.y));
204     });
205
206   // Draw the key & bind the click events
207    var key = svg.selectAll(".key")
208     .data(color.domain())
209     .enter().append("g")
210     .attr("class", "key")
211     .attr("transform", function(d, i) {
212       return "translate(0,"+i*25+")";}
213     );
214   key.append("rect")
215     .attr("x", 20)
216     .attr("y", 20)
217     .attr("width", 50)
218     .attr("height", 10)
219     .attr("line", function(d) {return d;})
220     .style("fill", color)
221     .on('click', function(d) {isolateLine(d)});
222   key.append("text")
223     .attr("x", 75)
224     .attr("y", 10)
225     .attr("dy", 20)
226     .text(function(d) { return d; });
227   }
228 }
229
230 // Toggle the isolation of a single metro line when clicked in the key
231 function isolateLine(d) {
232   var node = d3.selectAll(".node");
233   var link = d3.selectAll(".link");
234   // If no line is selected, isolate the line that was just clicked
235   if (iso_toggle==0) {
236     var edges = metro_lines[d];
237     node.transition().style("fill-opacity", function (o) {
238         return o.lines.indexOf(d) > -1 || isInterchange(o)? 1 : 0.1;
```

```
    }).style("stroke-opacity", function (o) {
        return o.lines.indexOf(d) > -1 ? 1 : 0.05;
    }).duration(700);
    link.transition().style("opacity", function (o) {
        return o.line == d? 1 : 0.1;
    }).duration(700);
  } else { // Reset everything
    node.transition().duration(700)
      .style("fill-opacity", 1)
      .style("stroke-opacity", 1);
    link.transition().duration(700)
      .style("opacity", 1);
  }
  force.start();
  force.tick(); // All the nodes are fixed so manually tick()
  force.stop();
  iso_toggle = 1-iso_toggle;
}

// Let the nodes move around a bit and then forceably pause them.

// To-do: polling until octilinearity/LS are suitably low, then doing
// force.alpha(0) breaks ALL THE THINGS. but until then we have to
// have a hard time limit which often performs badly.
function move(time) {
  for (var n=0; n<graph.nodes.length; n++) {
    graph.nodes[n].fixed = false;
  }
  force.start();
  window.setTimeout(function(){
  // Wait some completely arbitraty amount of time before 'pausing'
    force.alpha(0);
    for (var n=0; n<graph.nodes.length; n++) {
      graph.nodes[n].fixed = true;
    }
  }, time);
};

function getMetrics(graph, spacing) {
  var metrics = { x_min: width, x_max: 0,
                  y_min: height, y_max: 0,
                  x_avg: 0, y_avg: 0,
                  v_scale: 0, h_scale: 0 };

  metrics.x_min = Math.min.apply(
    null,graph.nodes.map(function(d){return d.x;}));
  metrics.y_min = Math.min.apply(
    null,graph.nodes.map(function(d){return d.y;}));
  metrics.x_max = Math.max.apply(
    null,graph.nodes.map(function(d){return d.x;}));
  metrics.y_max = Math.max.apply(
    null,graph.nodes.map(function(d){return d.y;}));

  metrics.x_avg = (metrics.x_min+metrics.x_max)/2.0;
  metrics.y_avg = (metrics.y_min+metrics.y_max)/2.0;

  metrics.v_scale = Math.abs(
    metrics.y_max-metrics.y_min)/(height-2*spacing
  );
  // Deliberately same denominator for v/h scales
  metrics.h_scale = Math.abs(
    metrics.x_max - metrics.x_min)/(height-2*spacing
  );
  metrics.x_move = (metrics.x_min*metrics.v_scale)/2.0;
  metrics.y_move = (metrics.y_min*metrics.h_scale)/2.0;

  return metrics;
}

function exportPositions() {
  positions = [];
  for (var n=0; n<graph.nodes.length; n++) {
    positions[n] = {"x": graph.nodes[n].x, "y": graph.nodes[n].y};
  }
  return JSON.stringify(positions);
}

function importPositions(positions) {
  for (var n=0; n<graph.nodes.length; n++) {
    graph.nodes[n].x = positions[n].x;
    graph.nodes[n].px = positions[n].x;
    graph.nodes[n].y = positions[n].y+10;
    graph.nodes[n].py = positions[n].y+10;
  }
  force.start();
  force.tick();
  force.stop();
}

function octilineariseLine(metro_line, begin, end, dir) {
```

```
  let l = metro_lines[metro_line];
  let line = {x: end.px-begin.px, y: end.py-begin.py};

  let alpha = Math.atan(line.y/line.x);
  let nearest = Math.ceil(alpha/(Math.PI/4)) * (Math.PI/4);

  if (dir == 1) { // line coming inwards; move 'begin' station
    if (Math.round(Math.abs(nearest),2)==2 || nan(Math.tan(nearest))) {
      begin.py += linelength(begin, end);
    } else {
      begin.py += line.x * (Math.tan(nearest)-Math.tan(alpha));
    }
  } else { // line going outwards; move 'end' station
    if (Math.round(Math.abs(nearest),2)==2 || nan(Math.tan(nearest))) {
      end.py += linelength(begin, end);
    } else {
      end.py += line.x * (Math.tan(nearest)-Math.tan(alpha));
    }
  }
  return {b:begin, e:end};
}

// start and stop are inclusive
function spaceAlongLine(l, start, stop, dir) {
  for (var n=0; n<graph.nodes.length; n++) {
    graph.nodes[n].px = graph.nodes[n].x;
    graph.nodes[n].py = graph.nodes[n].y;
  }
  var line = metro_lines[l];
  let real_begin = start==0?
    graph.nodes[line[0][0]]:
      graph.nodes[line[start-1][1]];
  let real_end = graph.nodes[line[stop-1][1]];
  oct = octilinear iseLine(l, real_begin, real_end, dir);
  let begin = oct.b;
  let end = oct.e;

  let delta = dist2d(begin, end);
  delta.x = Math.ceil(delta.x/(stop-start+1));
  delta.x = (Math.abs(3*delta.x) < spacing) ?
    delta.x*2: (Math.abs(delta.x)>spacing) ?
      delta.x/1.5 : delta.x;
  delta.y = Math.ceil(delta.y/(stop-start));
  delta.y = (Math.abs(3*delta.y) < spacing) ?
    delta.y*2: (Math.abs(delta.y)>spacing) ?
      delta.y/1.5 : delta.y;

  for (var s=start; s<=stop; s++) {
    let station = s==0?
      graph.nodes[line[0][0]]:
        graph.nodes[line[s-1][1]];
    station.x = begin.px + (s-start)*delta.x;
    station.y = begin.py + (s-start)*delta.y;
    station.px = station.x;
    station.py = station.y;
  }
}

// Snap nodes to the grid so that hopefully most of them are drawn
// octilinearly in a discrete point space, then do some basic line
// straightening along edges.
function snap(){
  force.stop();
  var metrics = getMetrics(graph, spacing);

  // This is the snap to grid part, we don't care about collisions
  // yet so don't mark anything as placed or unplaced
  for (var n=0; n<graph.nodes.length; n++) {
    let node = graph.nodes[n];
    node.x = (node.x/metrics.h_scale);
    node.y = (node.y/metrics.v_scale);
    c = { x: Math.ceil(node.x/spacing) * spacing,
          y: Math.ceil(node.y/spacing) * spacing};
    node.x = c.x; node.px = c.x;
    node.y = c.y; node.py = c.y;
  }

  metrics = getMetrics(graph, spacing);

  // We want to place stations in order of descending weight to keep
  // busy sections of the map as clean as possible
  var st = Object.keys(graph.nodes).sort(function(a,b){
    return graph.nodes[b].weight-graph.nodes[a].weight
  });
  // Try and place nodes if they can be snapped, otherwise nodes can
  // bump each other off if they are closer to the intersection
  for (var n of st) {
    let node = graph.nodes[n];
    node.x = (node.x-metrics.x_move) * (1.0/metrics.h_scale);
    node.y = (node.y-metrics.y_move) * (1.0/metrics.v_scale);
```

```
419
420     c = { x: Math.ceil((node.x)/spacing) * spacing,
421           y: Math.ceil((node.y)/spacing) * spacing};
422
423     if (!taken[coordinates(c)]){
424       taken[coordinates(c)] = node;
425       node.placed = true;
426       node.px = c.x;// node.px = c.x;
427       node.py = c.y;// node.py = c.y;
428     } else if (dist2d(node, c) < dist2d(taken[coordinates(c)], c)) {
429       let old = taken[coordinates(c)];
430       old.placed = false;
431       taken[coordinates(c)] = node;
432       node.placed = true;
433       node.px = c.x;// node.px = c.x;
434       node.py = c.y;// node.py = c.y;
435     }
436   }
437
438   for (var n=0; n<graph.nodes.length; n++) {
439     graph.nodes[n].px = graph.nodes[n].x;
440     graph.nodes[n].py = graph.nodes[n].y;
441   }
442
443   // If we have: (a)  x  (c) [ac is octilinear, b has no other links]
444   //               \     /
445   //                \   /
446   //                 (b)
447   // Transform this into: (a)--(b)--(c) [as long as x is not taken]
448
449   for (var line in metro_lines) {
450     line = metro_lines[line];
451     for (var s=1; s<line.length; s++) {
452       let a=graph.nodes[line[s-1][0]];
453       let b=graph.nodes[line[s][0]];
454       let c=graph.nodes[line[s][1]];
455       let candidate = {x:(a.px+c.px)/2.0, y:(a.py+c.py)/2.0};
456
457       if (octilinear(a, c) &&
458         b.weight <= 2 && !taken[coordinates(candidate)]){
459         taken[coordinates(b)] = false;
460         taken[coordinates(candidate)] = b;
461         b.x = candidate.x; b.px = candidate.x;
462         b.y = candidate.y; b.py = candidate.y;
463         b.placed = true;
464       }
465     }
466   }
467
468   for (var n=0; n<graph.nodes.length; n++) {
469     graph.nodes[n].px = graph.nodes[n].x;
470     graph.nodes[n].py = graph.nodes[n].y;
471   }
472
473   // If we have: (a)  x  (c) [ac is octilinear, b has no other links]
474   //               \     /
475   //                \   /
476   //                 (b)
477   // Transform this into: (a)--(b)--(c)
478   // [EVEN IF x IS TAKEN OR B IS PLACED ALREADY]
479
480   for (var line in metro_lines) {
481     line = metro_lines[line];
482     for (var s=line.length-1; s>0; s--) {
483       let a=graph.nodes[line[s][1]];
484       let b=graph.nodes[line[s-1][1]];
485       let c=graph.nodes[line[s-1][0]];
486       let candidate = {x:(a.px+c.px)/2.0, y:(a.py+c.py)/2.0};
487
488       if (octilinear(a, c) && b.weight == 2)  {
489         taken[coordinates(b)] = false;
490         taken[coordinates(candidate)] = b;
491         b.x = candidate.x; b.px = candidate.x;
492         b.y = candidate.y; b.py = candidate.y;
493         b.placed = true;
494       }
495     }
496   }
497
498   // If we have: (a)          [b has not been placed]
499   //               \
500   //                \
501   //                 (b)--(c)
502   //
503   // Transform this into: (a)
504   //                        \
505   //                        (b)
506   //                          \
507   //                          (c)
508   // [EVEN IF AC NOT OCTILINEAR OR B IS ON OTHER LINES]
```

```

  for (var line in metro_lines) {
  line = metro_lines[line];
    for (var s=1; s<line.length; s++) {
      let a=graph.nodes[line[s-1][0]];
      let b=graph.nodes[line[s][0]];
      let c=graph.nodes[line[s][1]];
      let candidate = {x:(a.px+c.px)/2.0, y:(a.py+c.py)/2.0};

      if (!b.placed)  {
        taken[coordinates(b)] = false;
        taken[coordinates(candidate)] = b;
        b.x, b.px = candidate.x, candidate.x;
        b.y, b.py = candidate.y, candidate.y;
        a.placed = true;
        b.placed = true;
        c.placed = true;
      }
    }
  }

  var to_space = {};
  // Forwards
  for (var l in metro_lines) {
    line = metro_lines[l];
    var s = 0;
    var start_from = graph.nodes[line[s][0]];
    if (start_from.weight==1) {
      var acc = graph.nodes[line[s][1]];
      while (acc.weight<=2 && s<line.length) {
        acc = graph.nodes[line[s][1]];
        ++s;
      }
      var end = s;
      if (end != 0) {
        to_space[l] = [[0, end, 1]];
      }
    }

    //Backwards
    s = line.length-1;
    start_from = graph.nodes[line[s][1]];
    if (start_from.weight==1) {
      var acc = graph.nodes[line[s][0]];
      while (acc.weight<=2 && s>0) {
        --s;
        acc = graph.nodes[line[s][0]];
      }
      var start = acc.weight<=2? s-1: s;
      if (start != line.length-1) {
        to_space[l] = to_space[l] || [];
        to_space[l].push([start, line.length, -1]);
      }
    }
  }
  for (line in to_space) {
    for (group of to_space[line]) {
      spaceAlongLine(line, group[0], group[1], group[2]);
    }
  }

  var metrics = getMetrics(graph, spacing);
  // We've moved the stations around a lot so the map probably needs
  // rescaling & recentering.
  // To-do, this suffers from the x/px problem, and NaNs sometimes.
  for (var n=0; n<graph.nodes.length; n++) {
    let node = graph.nodes[n];
    node.x = (node.x-metrics.x_move) * (1.0/metrics.h_scale);
    node.y = (node.y-metrics.y_move) * (1.0/metrics.v_scale);
    node.px = node.x;
    node.py = node.y;
  }

  // Headcount; how many stations couldn't we place?
  taken = {};
  for (var n=0; n<graph.nodes.length; n++) {
    if (taken[coordinates(graph.nodes[n])]) {
      taken[coordinates(graph.nodes[n])] = 1;
    } else {
      taken[coordinates(graph.nodes[n])] = 1;
    }
  }
  force.start();
  console.debug("octilinearity:", octilinearity());
  console.debug("lineStraightness:", lineStraightness());
}

var octilinearity = function(){
  var total = 0;
  for (var n=0; n<graph.links.length; n++) {
```

```
      let s = graph.links[n].source;
      let t = graph.links[n].target;
      if (s.py-t.py==0 || s.px-t.px==0) {
        continue; // implicit total += 0
      } else {
        let theta = 4*Math.atan(Math.abs((s.py-t.py)/(s.px-t.px)));
        total += Math.abs(Math.sin(theta));
      }
    }
  return total;
}

// This fails if any points are drawn on top of each other
var lineStraightness = function(){
  var total = 0;
  for (var line in metro_lines) {
    line = metro_lines[line];
    for (var s=0; s<line.length-1; s++) {
      let a = graph.nodes[line[s][0]];
      let b = graph.nodes[line[s+1][0]];
      let c = graph.nodes[line[s+1][1]];
      let v1 = {x:a.px - b.px, y:a.py - b.py};
      let v2 = {x:c.px - b.px, y:c.py - b.py};

      var v1mag = Math.sqrt(v1.x*v1.x + v1.y*v1.y);
      var v2mag =  Math.sqrt(v2.x*v2.x + v2.y*v2.y);
      var v1norm = {x:v1.x/v1mag, y:v1.y/v1mag};
      var v2norm = {x:v2.x/v2mag, y:v2.y/v2mag};

      var res = (v1norm.x * v2norm.x + v1norm.y * v2norm.y);
      // Make sure we don't NaN on e.g. acos(1.0000000000002)
      res = Math.round(res * 100) / 100;
      var theta = Math.acos(res);
      total+= theta;
    }
  }
  return total;
}

move(2000);
setTimeout(function(){snap();}, 2500);

// Function from http://stackoverflow.com/a/3177838
function timeSince(date) {
    var seconds = Math.floor((new Date() - date) / 1000);
    var interval = Math.floor(seconds / 31536000);
    if (interval > 1) { return interval + " years";}
    interval = Math.floor(seconds / 2592000);
    if (interval > 1) { return interval + " months"; }
    interval = Math.floor(seconds / 86400);
    if (interval > 1) { return interval + " days"; }
    interval = Math.floor(seconds / 3600);
    if (interval > 1) { return interval + " hours";}
    interval = Math.floor(seconds / 60);
    if (interval > 1) {return interval + " minutes";}
    return Math.floor(seconds) + " seconds";
}

function showArticle(data) {
  $('#article-container').get(0).innerHTML =
  "<h1>"+data.title+"</h1>"+
  "<h2>"+timeSince(data.date)+" ago</h2>"+ data.html;
}

$('#graphPane').click(function() {
  $("#cards").hide();
  $('#feedPane').removeClass("active");
  $("#graph").show();
  $("#article-container").show();
  $("body")[0].style.overflow = "hidden";
  $('#graphPane').addClass("active");
});

$("#feedPane").click(function() {
  $("#graph").hide();
  $("#article-container").hide();
  $('#graphPane').removeClass("active");
  $("#cards").show();
  $("body")[0].style.overflow = "scroll";
  $('#feedPane').addClass("active");
});

 $("#graphPane").trigger("click");
drawGraph(graph);
```

D.10 File: src/webUtils/newsgraph.css

```
body {
        overflow: none;
}

.header {
 padding: 5px;
 font-size: 2em;
}

h4 {
        font-size: 1.5em !important;
}

.nav-wrapper {
        z-index: 999 !important;
         position: relative;
         box-shadow: none;
}
.nav-wrapper:before {
  position: absolute;
  left: 0px;
  top: 0px;
  right: 0px;
  bottom: 0px;
  box-shadow: 0 2px 5px 0 rgba(0,0,0,0.16),
              0 2px 10px 0 rgba(0,0,0,0.12);
  pointer-events: none;
  content:"";
}

.node circle {
 stroke: #000;
 stroke-width: 4px;
}

.link {
  fill: none;
  stroke-width: 7px;
}

#cards {
        margin-top: 20px;
        overflow: scroll;
}

.card {
        margin: 10px;
}
.card img {
        height: 200px;
}
.card-title {
        font-size: 1.4em !important;
        line-height: 1.2em !important;
}

#graph {
        margin: 0px;
        border: 0px;
        overflow: none;
}

#article-container {
        position: fixed;
        z-index: 1 !important;
        top: 0px;
        bottom: 0px;
        right: 0px;
        width: 30%;
        padding: 70px 10px 10px;
  background: #eff0f1;
  border: 0px;
  margin:0px;
  box-shadow:-2px 2px 5px 0 rgba(0,0,0,0.16),
             -2px 2px 10px 0 rgba(0,0,0,0.12);
  overflow: scroll;
}
#article-container h1 {
        font-size: 2em;
        line-height: 110%;
}
#article-container h2 {
        font-size: 1.2em;
        font-style: italic;
}
```